Serial ATA

Advanced Host Controller Interface

(AHCI)

DRAFT

Revision 0.95.doc

Please send comments to Joe Bennett
joseph.a.bennett@intel.com

Table of Contents

Table of Figures

1 Introduction

Changes from previous version:

- Integrated the new chapter 5 and state machine (not done with change-bars).
- Added raw mode bit descriptions and behavior (with change bars).
- Updated overflow section showing when PxIS.OFS must (and may optionally) be set.
- Re-introduced an underflow section describing HBA behavior.
- Wrote the equations for when the LED is generated, replacing the text that was in section 12.10

1.1 Overview

AHCI describes a PCI class device that acts as an interface between system memory and SATA devices.

AHCI host devices (referred to as host bus adapters, or HBA) may support from 1 to 32 ports. An HBA must support ATA and ATAPI devices, and must support both the PIO and DMA protocols. The HBA may optionally support a command list on each port for overhead reduction, and to support SATA command queuing via the DMA Setup FIS protocol for each device of up to 32 entries. The HBA may optionally support 64-bit addressing.

AHCI describes a system memory structure which contains a generic area for control and status, and a table of entries describing a command list (an HBA which does not support a command list shall have a depth of one for this table). Each command list entry contains information necessary to program an SATA device, and a pointer to a descriptor table for transferring data between system memory and the device.

1.2 Scope

AHCI encompasses a PCI device. It contains a PCI BAR (Base Address Register) to implement native SATA features. AHCI contains definitions for the following features:

- Support for 32 ports
- Elimination of Master / Slave Handling
- Hot Plug
- Hardware Assisted Command Queuing
- Cold device presence detect
- Activity LED generation
- Port Multiplier
- 64-bit addressing
- Large LBA support
- Power Management
- Staggered Spin-up
- Serial ATA superset registers
- Port Selector

1.3 Outside of Scope

AHCI does not contain information relevant to implementing the transport, link or PHY layers of Serial ATA as this is wholly described in the SATA 1.0 specification. It does not describe enclosure management services, as these are covered in the Serial ATA II: Extensions to Serial ATA 1.0 specification. It also does not specify ATA legacy behavior, such as the legacy I/O ranges, or bus master IDE. Allowances have been made in AHCI so that an HBA may implement these features for backward compatibility with older operating systems (for example, the location of the memory BAR for AHCI is after the BAR locations for both native IDE and bus master IDE).

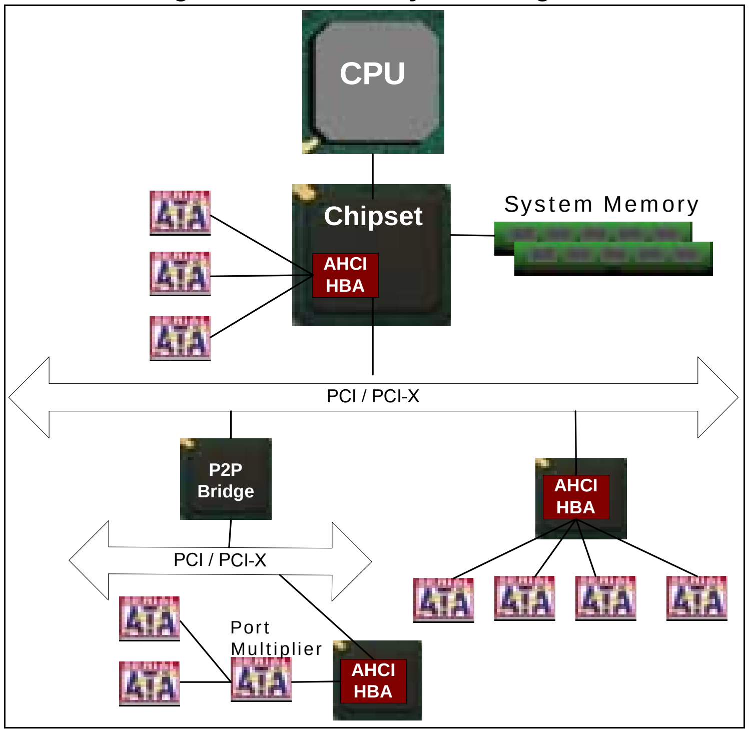
1.4 Block Diagram

In Figure 1, several AHCI HBAs are attached in an IA-based computer system. One HBA is integrated in the core chipset. Another sits off the first available PCI/PCI-X bus. (PCI is used as a reference name. The bus can be any PCI-like bus, such as PCI-X, PCI-Express, HT, etc.)

A final HBA sits off a second PCI bus that exists behind a PCI-PCI bridge. This last HBA has one port attached to a Port Multiplier

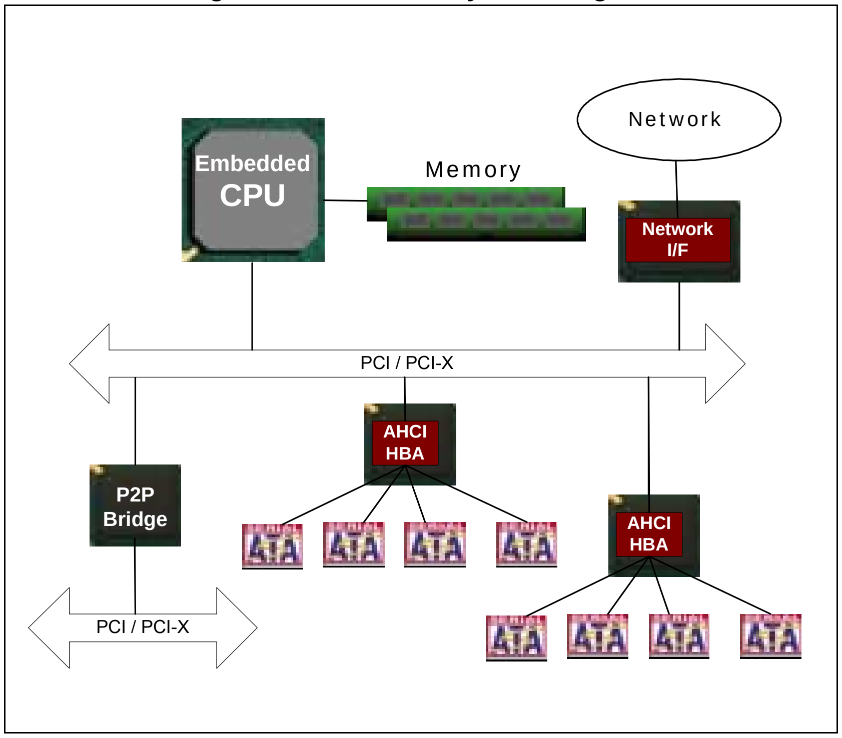
All the devices talk to system memory attached to the chipset.

Figure 1: IA Based System Diagram



In Figure 2, two HBA are connected to an embedded CPU with its own local memory. This configuration would most likely be used in a RAID-type environment,

Figure 2: Embedded System Diagram



1.5 Conventions

Hardware must return '0' for all bits and registers that are marked as reserved, and software must write all reserved bits and registers with the value of '0'.

Inside the register section, the following abbreviations are used:

RO	Read Only
RW	Read Write
RWC	Read/Write '1' to clear
RW1	Read/Write '1' to set
Impl. Spec.	Implementation Specific – the HBA has the freedom to choose its implementation.
HwInit	The default state is dependent initialized at reset, either by an expansion ROM, or in the case of integrated devices, by a platform BIOS.

When a register bit is referred to in the document, the convention used is "Register Symbol.Field Symbol". For example, the configuration space PCI command register parity error response bit is referred to by the name CMD.PEE. If the register field is an array of bits, the field will be referred to as "Register Symbol.Field Symbol(array offset)".

When a memory field is referred to in the document, the convention used is "Register Name[Offset Symbol]". For example, the pointer to the Command Header of port 0 is referred to by the name P0CLB[CH0].

1.6 Terminology

HBA	Host Bus Adapter – refers to the silicon that implements the AHCI specification to communicate between system memory and SATA devices.
H2D	HBA to Device
D2H	Device to HBA
System Memory	DRAM or "main" memory of a computer system, used to communicate data and command information between the host processor and the SATA device.
Register Memory	Registers allocated in the memory space of the HBA. These registers are physically implemented in the HBA.
Command List	Defines commands located in system memory that an HBA processes. This is a list that may be 1 to 32 entries called Command Slots, and may contain any type of ATA or ATAPI command. The command list is advanced when the BSY, DRQ, and ERR bits of an SATA device is cleared
Command Slot	One of the entries in the command list, which contains the command to execute. Up to 32 slots are supported in a Command List.
Queue	Indicates the specific ATA command queue inside an SATA device. This is differentiated from the command list in that a queue shall only exist in an SATA device when all the commands in the HBA's command list are of the SATA queued type
Port	A physical port on the HBA, with a set of registers that control the DMA and link operations. A physical port may have several devices attached to it via a Port Multiplier
Device	A physical device, such as an HDD or DVD that is either directly attached to an HBA port, or is attached through a Port Multiplier to an HBA port.

1.7 Theory of Operation

AHCI is defined to take the basics of the scatter/gather list concept of Bus Master IDE, and expand it to reduce CPU/software overhead and provide support for Serial ATA features such as hot plug, power management, and accessing of several devices without performing master/slave emulation.

Communication between a device and software moves from the task file via byte-wide accesses to a command FIS located in system memory that is fetched by the HBA. This reduces command setup time significantly, allowing for many more devices to be added to a single host controller. Software no longer communicates directly to a device via the task file.

AHCI is defined to keep the HBA relatively simple so that it can act as a data mover. An HBA implementing AHCI does not parse any of the ATA or ATAPI commands as they are transferred to the device, although it is not prohibited from doing so.

All data transfers between the device and system memory occur through the HBA acting as a bus master to system memory. Whether the transaction is of a DMA type or a PIO type, the HBA fetches and stores data to memory, offloading the CPU. There is no data port that can be accessed.

Even though all transfers are performed in a DMA fashion, the use of the PIO command type is strongly discouraged. PIO has limited support for errors – for example, the ending status field of a PIO transfer is given to the HBA during the PIO Setup FIS, before the data is transferred. However, some commands may only be performed via PIO commands (such as IDENTIFY_DEVICE). Therefore, limited support is available. Only single DRQ block transfers are allowed.

The AHCI defines a standard mechanism for implementing a SATA command queue using the DMA Setup FIS. An HBA which supports queuing has individual slots in the **Command List** allocated in system memory for all the commands. Software can place a command into any empty slot, and upon notifying the HBA via a register access, the HBA shall fetch the command and transfer it. The tag that is returned in the DMA Setup FIS is used as an index into the command list to get the scatter/gather list used in the transfer.

This command list can be used by system software and the HBA even when non-queued commands need to be transferred. System software can still place multiple commands in the list, whether DMA, PIO, or ATAPI, and the HBA shall walk the list and transfer them.

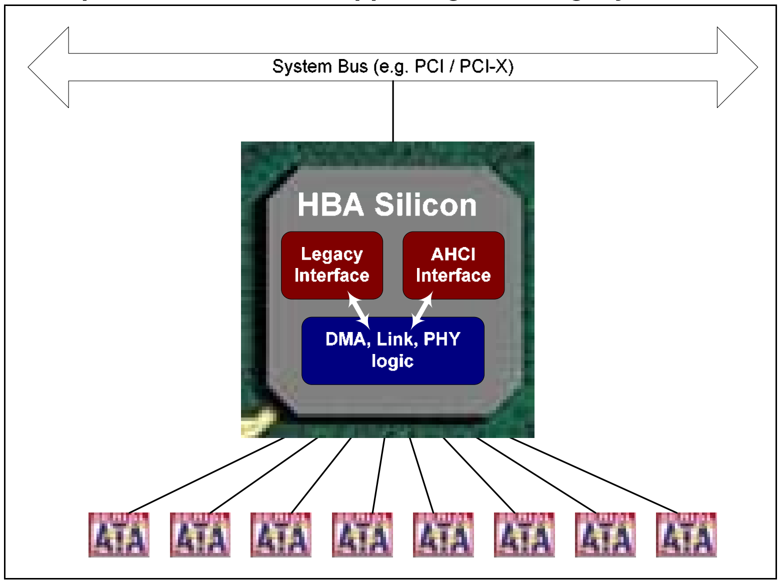
This multiple-use of the command list is achieved by the HBA only moving its command list pointer when the BSY, DRQ, and ERR bits are cleared by the device. It is system software's responsibility to not mix queued and non-queued commands in the command list.

1.8 Interaction with Legacy Software

AHCI is a self-contained specification that is intended to support all aspects of communicating with SATA devices, without having to utilize any legacy features such as shadow copies of the task file, snooping of bits in commands, etc.

HBAs that support legacy software mechanisms must do so in a fashion that is transparent to AHCI. Legacy registers are not allowed to affect any bits in AHCI registers, nor is AHCI software allowed to affect any bits in legacy registers. Software written for AHCI is not allowed to utilize any of the legacy mechanisms to program devices. In essence, an HBA that supports both mechanisms must isolate its legacy and AHCI engines, as shown in Figure 3.

Figure 3: Example of HBA Silicon Supporting Both Legacy and AHCI Interfaces



How an HBA that runs legacy software supports more than 4 ports is beyond the scope of this specification. How software transitions between legacy and AHCI modes of operation is beyond the scope of this specification.

1.9 References

The AHCI utilizes the following documents as references:

PCI Specification, Revision 2.3
- o http://www.pcisig.com

PCI Power Management Specification
- o http://www.pcisig.com

ATA/ATAPI-6
- o http://www.t13.org

Serial ATA 1.0a, Serial ATA II: Extensions to Serial ATA 1.0, Serial ATA II: Port Multiplier
- o http://www.serialata.org

Microsoft's Storage Device Class Power Management Specification:
- o http://www.microsoft.com/hwdev/resources/specs/pmref/default.asp

2 HBA Configuration Registers

Start (hex)	End (hex)	Name
00	3F	PCI Header
PMCAP	PMCAP+7	PCI Power Management Capability
MSICAP	MSICAP+9	Message Signaled Interrupt Capability

2.1 PCI Header

Start (hex)	End (hex)	Symbol	Name
00	03	ID	Identifiers
04	05	CMD	Command Register
06	07	STS	Device Status
08	08	RID	Revision ID
0A	0B	CC	Class Codes
0C	0C	CLS	Cache Line Size
0D	0D	MLT	Master Latency Timer
0E	0E	HTYPE	Header Type
0F	0F	BIST	Built In Self Test (Optional)
10	23	BARS	Other Base Address Registres (Optional) <BAR0-4>
24	27	ABAR	AHCI Base Address <BAR5>
2C	2F	SS	Subsystem Identifiers
30	33	EROM	Expansion ROM Base Address (Optional)
34	34	CAP	Capabilities Pointer
3C	3D	INTR	Interrupt Information
3E	3E	MGNT	Min Grant (Optional)
3F	3F	MLAT	Max Latency (Optional)

2.1.1 Offset 00h: ID - Identifiers

Bits	Type	Reset	Description
31:16	RO	Impl. Spec.	**Device ID (DID):** Indicates what device number assigned by the vendor.
15:00	RO	Impl. Spec.	**Vendor ID (VID):** 16-bit field which indicates the company vendor, assigned by the PCI SIG.

2.1.2 Offset 04h: CMD - Command

Bit	Type	Reset	Description
15:11	*RO*	*0*	*Reserved*
10	RW	0	**Interrupt Disable (ID):** Disables the HBA from generating interrupts.
09	RW	0	**Fast Back-to-Back Enable (FBE):** Allows the HBA to generate fast back-to-back cycles to different devices.
08	RW	0	**SERR# Enable (SEE):** When set, the HBA is allowed to generate SERR# on any event that is enabled for SERR# generation. When cleared, it is not.
07	*RO*	*0*	***Wait Cycle Enable (WCC):*** *Reserved.*
06	RW	0	**Parity Error Response Enable (PEE):** When set, the HBA shall generate PERR# when a data parity error is detected.
05	*RO*	*0*	***VGA Palette Snooping Enable (VGA):*** *Reserved*
04	RW	0	**Memory Write and Invalidate Enable (MWIE):** Allows the HBA to use the memory write and invalidate command.
03	*RO*	*0*	***Special Cycle Enable (SCE):*** *Reserved*
02	RW	0	**Bus Master Enable (BME):** Controls the HBA's ability to act as a master for data transfers. When this bit is cleared, HBA activity stops and any active DMA engines return to an idle condition. It is the equivalent of clearing the memory space start bits in each port.
01	RW	0	**Memory Space Enable (MSE)**: Controls access to the HBA's register memory space.
00	RW/ RO	Impl Spec	**I/O Space Enable (IOSE):** Controls access to the HBA's target I/O space. If the HBA also supports bus master IDE, this bit must be read/write. For a native AHCI implementation, this bit is read-only.

2.1.3 Offset 06h: STS - Device Status

Bit	Type	Reset	Description
15	RWC	0	**Detected Parity Error (DPE):** Set when the HBA detects a parity error on its interface.
14	RWC	0	**Signaled System Error (SSE):** Set when the HBA host generates SERR#.
13	RWC	0	**Received Master-Abort (RMA):** Set when the HBA receives a master abort to a cycle it generated.
12	RWC	0	**Received Target Abort (RTA):** Set when the HBA receives a target abort to a cycle it generated.
11	RWC	0	**Signaled Target-Abort (STA):** Set when the HBA terminates with a target abort.
10:09	RO	Impl. Spec.	**DEVSEL# Timing (DEVT):** Controls the device select time for the HBA's PCI interface.
08	RO	0	**Master Data Pariy Error Detected (DPD):** Set when the HBA, as a master, either detects a parity error or sees the parity error line asserted, and the parity error response bit (bit 6 of the command register) is set.
07	RO	Impl. Spec.	**Fast Back-to-Back Capable (FBC):** Indicates whether the HBA can accept fast back-to-back cycles.
06	*RO*	*0*	*Reserved*
05	RO	Impl. Spec.	**66 MHz Capable (C66):** Indicates whether the HBA can operate at 66 MHz.
04	RO	1	**Capabilities List (CL):** Indicates the presence of a capabilities list. The HBA must support the PCI power management capability as a minimum.
03	RO	0	**Interrupt Status (IS):** Indicates the interrupt status of the device (1 = asserted).
02:00	*RO*	*0*	*Reserved*

2.1.4 Offset 08h: RID - Revision ID

Bits	Type	Reset	Description
07:00	RO	00h	**Revision ID (RID):** Indicates stepping of the HBA hardware.

2.1.5 Offset 0Ah: CC - Class Code

Bits	Type	Reset	Description
23:16	RO	01h	**Base Class Code (BCC):** Indicates that this is a mass storage device.
15:08	RO	06h	**Sub Class Code (SCC):** Indicates that this is a Serial ATA device.
07:00	RO	01h	**Programming Interface (PI):** Indicates that this is an AHCI 1.0 HBA.

2.1.6 Offset 0Ch: CLS – Cache Line Size

Bits	Type	Reset	Description
07:00	RW	00h	**Cache Line Size (CLS):** Indicates the cache line size for use with the memory write and invalidate command, and as an indication on when to use the memory read multiple, memory read line, or memory read commands.

2.1.7 Offset 0Dh: MLT – Master Latency Timer

Bits	Type	Reset	Description
07:00	RW	00h	**Master Latency Timer (MLT):** Indicates the number of clocks the HBA is allowed to act as a master on PCI.

2.1.8 Offset 0Eh: HTYPE – Header Type

Bits	Type	Reset	Description
07	RO	Impl Spec	**Multi-Funciton Device (MFD):** Indicates whether the HBA is part of a multi-function device.
06:00	RO	00h	**Header Layout (HL):** Indicates that the HBA uses a target device layout.

2.1.9 Offset 0Fh: BIST – Built In Self Test (Optional)

The following register is optional, but if implemented, must look as follows.

Bits	Type	Reset	Description
07	RO	1	**BIST Capable (BC):** Indicates whether the HBA has a BIST function. This does not indicate SATA BIST capability – this is only for an HBA related BIST function.
06	RW	0	**Stat BIST (SB):** Software sets this bit to invoke BIST. The HBA clears this bit when BIST is complete.
05:04	*RO*	*00*	*Reserved*
03:00	RO	0h	**Completion Code (CC):** Indicates the completion code status of BIST. A non-zero value indicates a failure.

2.1.10 Offset 10h – 20h: BARS – Other Base Addresses (Optional)

These registers allocate memory or I/O spaces for other BARs. An example application of these BARs is to implement the native IDE and bus master IDE ranges for an HBA that wishes to support legacy software.

2.1.11 Offset 24h: ABAR – AHCI Base Address

This register allocates space for the HBA memory registers defined in section 3.

Bit	Type	Reset	Description
31:13	RW	0	**Base Address (BA):** Base address of register memory space. This represents a memory space for support of 32 ports. For HBAs that support less than 32-ports, more bits are allowed to be R/W, and therefore less memory space is consumed.
12:04	*RO*	*0*	*Reserved*
03	RO	0	**Prefetchable (PF):** Indicates that this range is not pre-fetchable
02:01	RO	00	**Type (TP):** Indicates that this range can be mapped anywhere in 32-bit address space
00	RO	0	**Resource Type Indicator (RTE):** Indicates a request for register memory space.

2.1.12 Offset 2Ch: SS - Sub System Identifiers

Bits	Type	Reset	Description
31:16	RO	0000h	**Subsystem ID (SSID):** Indicates the sub-system identifier.
15:00	RO	0000h	**Subsystem Vendor ID (SSVID):** Indicates the sub-system vendor identifier

2.1.13 Offset 30h: EROM – Expansion ROM (Optional)

Bit	Type	Reset	Description
31:00	RW	Impl Spec	**ROM Base Address (RBA):** Indicates the base address of the HBA expansion ROM..

2.1.14 Offset 34h: CAP – Capabilities Pointer

Bit	Type	Reset	Description
7:0	RO	PMCAP	**Capability Pointer (CP):** Indicates the first capability pointer offset. It points to the PCI Power management capability offset.

2.1.15 Offset 3Ch: INTR - Interrupt Information

Bits	Type	Reset	Description
15:08	RO	01h	**Interrupt Pin (IPIN):** This indicates that the HC generates the INTA# pin.
07:00	RW	00h	**Interrupt Line (ILINE):** Software written value to indicate which interrupt line (vector) the interrupt is connected to. No hardware action is taken on this register.

2.1.16 Offset 3Eh: MGNT – Minimum Grant (Optional)

Bits	Type	Reset	Description
07:00	RO	Impl Spec	**Grant (GNT):** Indicates the minimum grant time (in ¼ microseconds) that the device wishes grant asserted.

2.1.17 Offset 3Fh: MLAT – Maximum Latency (Optional)

Bits	Type	Reset	Description
07:00	RO	Impl Spec	**Latency (LAT): I**ndicates the maximum latency (in ¼ microseconds) that the device can withstand.

2.2 PCI Power Management Capabilities

Start (hex)	End (hex)	Symbol	Name
PMCAP	PMCAP+1	PID	PCI Power Management Capability ID
PMCAP+2	PMCAP+3	PC	PCI Power Management Capabilities
PMCAP+4	PMCAP+7	PMCS	PCI Power Management Control and Status

2.2.1 Offset PMCAP: PID - PCI Power Management Capability ID

Bit	Type	Reset	Description
15:08	RO	MSICAP	**Next Capability (NEXT):** Indicates the location of the next item in the list is the MSI capability.
07:00	RO	01h	**Cap ID (CID):** Indicates that this pointer is a PCI power management.

2.2.2 Offset PMCAP + 2h: PC – PCI Power Management Capabilities

Bit	Type	Reset	Description
15:11	RO	Impl Spec	**PME_Support (PSUP):** Indicates the states that can generate PME#. The states that can cause a PME for an HBA are implementation specific, and may be '0' (no states). If the HBA supports interlock switches (memory space CAP.SIS set) or cold presence detect (memory space CAP.SCD), then PME# must be supported from D3$_{HOT}$ (D3$_{COLD}$ may still be optional). The encoding of this field is as follows: **Bit** – **State** 15 – When set, PME# can be generated from D3$_{COLD}$. When cleared, PME# cannot be generated from D3$_{COLD}$. 14 – When set, PME# can be generated from D3$_{HOT}$. When cleared, PME# cannot be generated from D3$_{HOT}$. 13 – This bit must be '0' for AHCI HBAs. D2 is not a supported HBA state 12 – This bit must be '0' for AHCI HBAs. D1 is not a supported HBA state. 11 – This bit must be '0' for AHCI HBAs. PME# from D0 is not supported – interrupts are used in D0.
10	RO	0	**D2_Support (D2S):** The D2 state is not supported for AHCI HBAs.
09	RO	0	**D1_Support (D1S):** The D1 state is not supported for AHCI HBAs.
08:06	RO	111	**Aux_Current (AUXC):** Reports 375mA maximum Suspend well current required when in the D3$_{COLD}$ state.
05	RO	0	**Device Specific Initialization (DSI):** Indicates that no device-specific initialization is required.
04	*RO*	*0*	*Reserved*
03	RO	0	**PME Clock (PMEC):** Indicates that PCI clock is not required to generate PME#.
02:00	RO	010	**Version (VS):** Indicates support for Revision 1.1 of the *PCI Power Management Specification.*

2.2.3 Offset PMCAP + 4h: PMCS – PCI Power Management Control And Status

Bit	Type	Reset	Description
15	RWC	Impl Spec	**PME Status (PMES):** Set when the HBA generates PME#. If the HBA implementation does not support generating PME# (PMCAP.PSUP field is all ‘0’), then this bit may be implemented as RO. If the HBA supports waking from D3$_{COLD}$, this bit is indeterminate at system boot. If the HBA does not support waking from D3$_{COLD}$, this bit is ‘0’ at system boot.
14:09	*RO*	*0*	*Reserved – AHCI HBA does not implement the data register.*
08	RW	Impl Spec	**PME Enable (PMEE):** When set, the HBA asserts the PME# signal when PMES is set. If the HBA does not support generating PME# (PMCAP.PSUP field is ‘0’), then this bit may be implemented as RO, defaulting to ‘0’. If the HBA supports waking from D3$_{COLD}$, this bit is indeterminate at system boot. If the HBA does not support waking from D3$_{COLD}$, this bit is ‘0’ at system boot.
07:02	*RO*	*0*	*Reserved*
01:00	R/W	00	**Power State (PS):** This field is used both to determine the current power state of the HBA and to set a new power state. The values are: 00 – D0 state 11 – D3$_{HOT}$ state The D1 and D2 states are not supported for AHCI HBAs. When in the D3$_{HOT}$ state, the HBA’s configuration space is available, but the register memory spaces are not. Additionally, interrupts are blocked.

2.3 Message Signaled Interrupt Capability

Start (hex)	End (hex)	Symbol	Name
MSICAP	MSICAP+1	MID	Message Signaled Interrupt Capability ID
MSICAP+2	MSICAP+3	MC	Message Signaled Interrupt Message Control
MSICAP+4	MSICAP+7	MA	Message Signaled Interrupt Message Address
MSICAP+8	MSICAP+9	MD	Message Signaled Interrupt Message Data

2.3.1 Offset MSICAP: MID – Message Signaled Interrupt Identifiers

Bits	Type	Reset	Description
15:08	RO	Impl Spec.	**Next Pointer (NEXT):** Indicates the next item in the list. This can be other capability pointers (such as PCI-X or PCI-Express) or it can be the last item in the list.
07:00	RO	05h	**Capability ID (CID):** Capabilities ID indicates MSI.

2.3.2 Offset MSICAP + 2h: MC – Message Signaled Interrupt Message Control

Bits	Type	Reset	Description
15:08	*RO*	*0*	*Reserved*
07	RO	0	**64 Bit Address Capable (C64):** Capable of generating a 32-bit message only.
06:04	RW	000	**Multiple Message Enable (MME):** Indicates the number of messages the HBA should assert. See section 12.6.2. If the value programmed into this field exceeds the MMC field in this register, only a single message shall be generated.
03:01	RO	HwInit	**Multiple Message Capable (MMC):** Indicates the number of messages the HBA wishes to assert. See section 12.6.2.
00	RW	0	**MSI Enable (MSIE):** If set, MSI is enabled and traditional interrupt pins are not used to generate interrupts.

2.3.3 Offset MSICAP + 4h: MA – Message Signaled Interrupt Message Address

Bits	Type	Reset	Description
31:02	RW	0	**Address (ADDR):** Lower 32 bits of the system specified message address, always DW aligned.
01:00	*RO*	*00*	*Reserved*

2.3.4 Offset MSICAP + 8h: MD – Message Signaled Interrupt Message Data

Bits	Type	Reset	Description
15:00	RW	0	**Data (Data):** This 16-bit field is programmed by system software if MSI is enabled. Its

			content is driven onto the lower word (PCI AD[15:0]) during the data phase of the MSI memory write transaction.

2.4 Other Capability Pointers

Though not mentioned in this specification, other capability pointers may be necessary, depending upon the implementation space. Examples would be the PCI-X capability for PCI-X implementations, PCI-Express capability for PCI-Express implementations, and potentially the vendor specific capability pointer.

These capabilities are beyond the scope of this specification.

3 HBA Memory Registers

The memory mapped registers within the HBA exist in non-cacheable memory space. Additionally, locked accesses are not supported. If software attempts to perform locked transactions to the registers, indeterminate results may occur.

The registers are broken into 2 sections – generic host control and port control. The port control registers are the same for all ports, and there are as many register banks as there are ports.

All registers not defined and all reserved bits within registers return '0' when read.

Start	End	Description
00	1F	Generic Host Control: These are registers that apply to the entire HBA.
20	*FF*	*Reserved*
100	17F	Port 0 port control registers
180	1FF	Port 1 port control registers
200	FFF	(Ports 2 – port 29 control registers)
1000	107F	Port 30 port control registers
1080	10FF	Port 31 port control registers

3.1 Generic Host Control

The following registers apply to the entire HBA.

Start	End	Symbol	Description
00	03	CAP	Host Capabilities
04	07	GHC	Global Host Control
08	0B	IS	Interrupt Status
0C	0F	PI	Ports Implemented
10	13	VS	Version

3.1.1 Offset 00h: CAP – HBA Capabilities

This register indicates basic capabilities of the HBA to driver software.

Bit	Type	Reset	Description
31	RO	Impl Spec	**Supports 64-bit Addressing (S64A):** Indicates whether the HBA can access 64-bit data structures. If true, the HBA shall make the 32-bit upper bits of the port DMA Descriptor, the PRD Base, and each PRD entry read/write. If cleared, these are read-only and treated as '0' by the HBA.
30	RO	Impl Spec	**Supports Command Queue Acceleration (SCQA):** Indicates whether the HBA supports SATA command queuing via the DMA Setup FIS. If set to '1', an HBA shall handle DMA Setup FISes natively, and can handle auto-activate optimization through that FIS. If cleared to '0', command queuing is not supported.
29	RO	HwInit	**Supports Cold Presence Detect (SCD):** Indicates whether the HBA is in a platform that supports cold presence detect. A platform that accomodates cold presence detect requires extra logic on the backplane or host board. This value is loaded by the expansion ROM or platform BIOS prior to OS initialization. HBAs that support cold presence detect must have an additional input pin per port to capture the status of the voltage comparitor.
28	RO	HwInit	**Supports Interlock Switch (SIS):** Indicates whether the HBA supports interlock switches on its ports for use in hot plug operations. This value is loaded by the expansion ROM or platform BIOS prior to OS initialization. HBAs that support interlock switches must have one extra pin per port.
27	RO	HwInit	**Supports Staggered Spin-up (SSS):** Indicates whether the HBA supports staggered spin-up on its ports, for use in balancing power spikes. This value is loaded by the expansion ROM or platform BIOS prior to OS initiallization.
26	RO	Impl. Spec	**Supports Aggressive Link Power Management (SALP):** Indicates that the HBA can support auto-generating link requests to the partial or slumber states when there are no commands to process.
25	RO	Impl. Spec	**Supports Activity LED (SAL):** Indicates that the HBA supports a single output pin which indicates activity. This pin can be connected to an LED on the platform to indicate device activity. See section 12.10 for more information.
24	RO	Impl. Spec	**Supports Raw FIS Mode (SRM):** When set, indicates that the HBA supports a RAW FIS mode of operation. For details on this mode of operation, see section TBD.
23:20	RO	Impl. Spec	**Interface Speed Support (ISS):** Indicates the maximum speed this HBA can support on its ports. These enodings match the PxSCTL.DET field, which is programmable by system software. Values are: **Bits** – **Definition** 0000 – No speed restrictions 0001 – Gen 1 (1.5 Gbps) 0010 – Gen 1 (1.5 Gbps) and Gen 2 (3 Gbps) *0011 - 1111* – *Reserved*
19	RO	0	**Supports Non-Zero DMA Offsets (SNZO):** Indicates whether the HBA can support non-zero DMA offsets for First Party DMA FISes. This bit is reserved for future AHCI enhancements. First generation HBAs must have this bit cleared to '0'.
18	RO	Impl. Spec	**Supports Port Selector Accleration (PSSA):** Indicates whether the HBA can support accleration features of a Port Selector (COMRESET sequence). When cleared, acceleration features are not supported. When set, acceleration features are supported.
17	RO	Impl. Spec	**Supports Port Multiplier (PMS):** Indicates whether the HBA can support a Port Multiplier. When set, a Port Multiplier is supported. If set, the PMFS bit in this register indicates whether command-based switching (PMFS = 0) or command-based and FIS-based switching (PMFS = 1) is supported. When cleared, a Port Multiplier is not supported, and a Port Multiplier may not be attached to this HBA. If this bit is cleared, the PMFS bit in this register is ignored.

Bit	Type	Reset	Description
16	RO	0	**Supports Port Multiplier FIS Based Switching (PMFS):** If the PMS bit in this register is set, this bit indicates whether the HBA can support FIS-based switching when a Port Multiplier is attached to a port. This bit is reserved for future AHCI enhancements. First generation HBAs must have this bit cleared to '0'.
15	*RO*	*0*	*Reserved*
14	RO	Impl Spec.	**Slumber State Capable (SSC):** Indicates whether the HBA can support transitions to the slumber state. When cleared, software must not allow the HBA to initiate transitions to the slumber state via agressive link power management nor the PxCMD.ICC field in each port, and the PxSCTL.IPM field in each port must be programmed to disallow device initiated slumber requests. When set, HBA and device initiated slumber requests can be supported.
13	RO	Impl Spec.	**Partial State Capable (PSC):** Indicates whether the HBA can support transitions to the partial state. When cleared, software must not allow the HBA to initiate transitions to the partial state via agressive link power management nor the PxCMD.ICC field in each port, and the PxSCTL.IPM field in each port must be programmed to disallow device initiated partial requests. When set, HBA and device initiated partial requests can be supported.
12:08	RO	Impl. Spec.	**Number of Command Slots (NCS):** 0's based value indicating the number of command slots supported by this HBA. A minimum of 1 and maximum of 32 slots can be supported.
07:05	*RO*	*0*	*Reserved*
04:00	RO	Impl. Spec.	**Number of Ports (NP):** 0's based value indicating the number of ports supported. A maximum of 32 ports can be supported. A value of '0h', indicating one port, is the minimum requirement.

3.1.2 Offset 04h: GHC – Global HBA Control

This register controls various global actions of the HBA.

Bit	Type	Reset	Description
31	RW	0	**AHCI Enable (AE):** When set, indicates that an AHCI driver is loaded and communication to the HBA shall be via AHCI mechanisms. This can be used by an HBA that supports both legacy mechanisms (such as SFF-8038i) and AHCI to know when the HBA is running under an AHCI driver. When set, software shall only communicate with the HBA using AHCI. The HBA does not have to allow command processing via both AHCI and legacy mechanisms. When cleared, software shall only communicate with the HBA using legacy mechanisms. An HBA may ignore this bit – it is here to enable easier HBA validation.
30:02	*RO*	*0*	*Reserved*
01	RW	0	**Interrupt Enable (IE):** This global bit enables interrupts from the HBA. When cleared (reset default), all interrupt sources from all ports are disabled. When set, interrupts are enabled.
00	RW1	0	**HBA Reset (HR):** When set by SW, this bit causes an internal reset of the HBA. All state machines that relate to data transfers and queuing shall return to an idle condition, and all ports shall be re-initialized via COMRESET. When the HBA has performed the reset action, it shall reset this bit to '0'. A software write of '0' shall have no effect. For a description on which bits are reset when this bit is set, see section 12.3.3.

3.1.3 Offset 08h: IS – Interrupt Status Register

This register indicates which of the ports within the controller have an interrupt pending and require service.

Bit	Type	Reset	Description
31:0	RWC	0	**Interrupt Pending Status (IPS):** If set, indicates that the corresponding port has an interrupt pending. Software can use this information to determine which ports require service after an interrupt. Only the ports that are implemented have a corresponding bit – all other bits are reserved.

3.1.4 Offset 0Ch: PI – Ports Implemented

This register indicates which ports are exposed to the HBA. It is loaded by an expansion ROM or platform BIOS. It indicates which ports that the device supports are available for software to use. Any available port may not be implemented. For example, on a device that supports 6 ports, only ports 1 and 3 could be available, with ports 0, 2, 4, and 5 being unavailable.

For ports that are not implemented, software must not write to registers within the port, and the HBA returns '0's on all reads to the registers.

The intent of this register is to allow system vendors to build platforms that support less than the full number of ports possible on the HBA silicon.

Bit	Type	Reset	Description
31:0	RO	HwInit	**Port Implemented (PI):** If set, the port is available for use. If cleared, the port is not available for use.

3.1.5 Offset 10h: VS – AHCI Version

This register indicates the major and minor version of the AHCI specification. It is BCD encoded. The upper two bytes represent the major version number, and the lower two bytes represent the minor version number. Example: Version 3.12 would be represented as 00030102h. Two versions of the specification are supported: 0.95, and 1.0.

3.1.5.1 VS Value for 0.95 Compliant HBAs

Bit	Type	Reset	Description
31:00	RO	0000h	**Major Version Number (MJR):** Indicates the major version is "0"
15:00	RO	0905h	**Minor Version Number (MNR):** Indicates the minor version is "95".

3.1.5.2 VS Value for 1.0 Compliant HBAs

Bit	Type	Reset	Description
31:00	RO	0001h	**Major Version Number (MJR):** Indicates the major version is "1"
15:00	RO	0000h	**Minor Version Number (MNR):** Indicates the minor version is "0".

3.2 Port Registers (one set per port)

The following registers describe the registers necessary to implement port 0. Additional ports shall have the same register mapping. Port 1 starts at 180h, port 2 starts at 200h, port 3 at 280h, etc. The algorithm for software to determine the offset is as follows:

- Port offset = 100h + (PI Asserted Bit Position * 80h)

Start	End	Symbol	Description
100	103	P0CLB	Port 0 Command List Base Address
104	107	P0CLBU	Port 0 Command List Base Address Upper 32-Bits
108	10B	P0FB	Port 0 FIS Base Address
10C	10F	P0FBU	Port 0 FIS Base Address Upper 32-Bits
110	113	P0IS	Port 0 Interrupt Status
114	117	P0IE	Port 0 Interrupt Enable
118	11C	P0CMD	Port 0 Command
120	123	P0TFD	Port 0 Task File Data
124	127	P0SIG	Port 0 Signature
128	12B	P0SSTS	Port 0 Serial ATA Status (SCR0: SStatus)
12C	12F	P0SCTL	Port 0 Serial ATA Control (SCR2: SControl)
130	133	P0SERR	Port 0 Serial ATA Error (SCR1: SError)
134	137	P0SACT	Port 0 Serial ATA Active (SCR3: SActive)
138	13B	P0CI	Port 0 Command Issue
140	143	P0RMCS	Port 0 Raw Mode Control and Status

3.2.1 Offset 100h: P0CLB – Port 0 Command List Base Address

Bit	Type	Reset	Description
31:10	RW	Impl Spec	**Command List Base Address (CLB):** Indicates the 32-bit base physical address for the command list for this port. This base is used when fetching commands to execute. The structure pointed to by this address range is 1K-bytes in length. This address must be 1K-byte aligned as indicated by bits 09:00 being read only. The size of the structure is 1K-bytes, so a 1K-byte alignment allows simple hardware processing.
09:00	*RO*	*0*	*Reserved*

3.2.2 Offset 104h: P0CLBU – Port 0 Command List Base Address Upper 32-bits

Bit	Type	Reset	Description
31:00	RW/ RO	Impl Spec	**Command List Base Address Upper (CLBU):** Indicates the upper 32-bits for the command list base physical address for this port. This base is used when fetching commands to execute. This register shall be read only ‘0’ for HBAs that do not support 64-bit addressing.

3.2.3 Offset 108h: P0FB – Port 0 FIS Base Address

Bit	Type	Reset	Description
31:08	RW	Impl Spec	**FIS Base Address (FB):** Indicates the 32-bit base physical address for received FISes. The structure pointed to by this address range is 256 bytes in length. This address must be 256-byte aligned as indicated by bits 07:00 being read only. The size of the structure is 256-bytes, so a 256-byte alignment allows simple hardware processing.
07:00	*RO*	*0*	*Reserved*

3.2.4 Offset 10Ch: P0FBU – Port 0 FIS Base Address Upper 32-bits

Bit	Type	Reset	Description
31:00	RW/ RO	Impl Spec	**Command List Base Address Upper (CLBU):** Indicates the upper 32-bits for the received FIS base physical address for this port. This register shall be read only ‘0’ for HBAs that do not support 64-bit addressing.

3.2.5 Offset 110h: P0IS – Port 0 Interrupt Status

Bit	Type	Reset	Description
31	RWC	0	**Cold Port Detect Status (CPDS):** When set, a device status has changed as detected by the cold presence detect logic. This bit can either be set due to a non-connected port receiving a device, or a connected port having its device removed. This bit is only valid in systems that support cold device connect (CAP.SCD set).
30	RWC	0	**Task File Error Status (TFES):** This bit is set whenever the status register is updated by the device and the error bit (bit 0) is set.
29	RWC	0	**Host Bus Fatal Error Status (HBFS):** Indicates that the HBA encountered a host bus error that it cannot recover from, such as a bad software pointer. In PCI, such an indication would be a target or master abort.
28	RWC	0	**Host Bus Data Error Status (HBDS):** Indicates that the HBA encountered a data error (uncorrectable ECC / parity) when reading from or writing to system memory.
27	RWC	0	**Interface Fatal Error Status (IFS):** Indicates that the HBA encountered an error on the SATA interface which caused the transfer to stop.
26	RWC	0	**Interface Non-fatal Error Status (INFS):** Indicates that the HBA encountered an error on the SATA interface but was able to continue operation.
25	*RO*	*0*	*Reserved*
24	RWC	0	**Overflow Status (OFS):** Indicates that the HBA received more bytes from a device than was specified in the PRD table for the command.
23	RWC	0	**Incorrect Port Multiplier Status (IPMS):** Indicates that the HBA received a FIS from a device whose Port Multiplier field did not match what was expected.
22:08	*RO*	*0*	*Reserved*
07	RWC	0	**Device Interlock Status (DIS):** When set, indicates that a platform interlock switch has been opened or closed, which may lead to a change in the connection state of the device. This bit is only valid in systems that support an interlock switch (CAP.SIS set). For systems that do not support an interlock switch, this bit shall always be '0'.
06	RO	0	**Port Connect Change Status (PCS):** 1=Change in *Current Connect Status*. 0=No change in *Current Connect Status*. This bit reflects the state of PxSERR.DIAG.X. Unlike other bits in this register, this bit is only cleared when PxSERR.DIAG.X is cleared.
05	RWC	0	**Descriptor Processed (DPS):** A PRD with the 'I' bit set has transferred all of its data.
04	RO	0	**Unknown FIS Interrupt (US):** An unknown FIS was received and has been copied into system memory. This bit reflects the state of PxSERR.DIAG.F. Unlike other bits in this register, this bit is only cleared when PxSERR.DIAG.F is cleared.
03	RWC	0	**Set Device Bits Interrupt (SDBS):** A Set Device Bits FIS has been received with the 'I' bit set and has been copied into system memory.
02	RWC	0	**DMA Setup FIS Interrupt (DSS):** A DMA Setup FIS has been received with the 'I' bit set and has been copied into system memory.
01	RWC	0	**PIO Setup FIS Interrupt (PSS):** A PIO Setup FIS has been received with the 'I' bit set, it has been copied into system memory, and the data related to that FIS has been transferred. This bit shall be set even if the data transfer resulted in an error.
00	RWC	0	**Device to Host Register FIS Interrupt (DHRS):** A D2H Register FIS has been received with the 'I' bit set, and has been copied into system memory.

3.2.6 Offset 114h: P0IE – Port 0 Interrupt Enable

This register enables and disables the reporting of the corresponding interrupt to system software. When a bit is set ('1') and the corresponding interrupt condition is active, then an interrupt is generated. Interrupt sources that are disabled ('0') are still reflected in the status registers. This register is symmetrical with the P0IS register.

Bit	Type	Reset	Description
31	RW/ RO	0	**Cold Presence Detect Enable (CPDE):** When set, GHC.IE is set, and P0S.CPDS is set, the HBA shall generate an interrupt. For systems that do not support cold presence detect, this bit shall be a read-only '0'.
30	RW	0	**Task File Error Enable (TFEE):** When set, GHC.IE is set, and P0S.TFES is set, the HBA shall generate an interrupt.
29	RW	0	**Host Bus Fatal Error Enable (HBFE):** When set, GHC.IE is set, and P0IS.HBFS is set, the HBA shall generate an interrupt.
28	RW	0	**Host Bus Data Error Enable (HBDE):** when set, GHC.IE is set, and P0IS.HBDS is set, the HBA shall generate an interrupt..
27	RW	0	**Interface Fatal Error Enable (IFE):** When set, GHC.IE is set, and P0IS.IFS is set, the HBA shall generate an interrupt..
26	RW	0	**Interface Non-fatal Error Enable (INFE):** When set, GHC.IE is set, and P0IS.INFS is set, the HBA shall generate an interrupt.
25	*RO*	*0*	*Reserved*
24	RW	0	**Overflow Enable (OFE):** When set, and GHC.IE and P0IS.OFS are set, the HBA shall generate an interupt.
23	RW	0	**Incorrect Port Multiplier Enable (IPME):** When set, and GHC.IE and P0IS.IPMS are set, the HBA shall generate an interupt.
22:08	*RO*	*0*	*Reserved*
07	RW/ RO	0	**Device Interlock Enable (DIE):** When set, and P0IS.DIS is set, the HBA shall generate an interrupt. For systems that do not support an interlock switch, this bit shall be a read-only '0'.
06	RW	0	**Port Change Interrupt Enable (PCE):** When set, GHC.IE is set, and P0IS.PCS is set, the HBA shall generate an interrupt.
05	RW	0	**Descriptor Processed Interrupt Enable (DPE):** When set, GHC.IE is set, and P0IS.DPS is set, the HBA shall generate an interrupt.
04	RW	0	**Unknown FIS Interrupt Enable (UE):** When set, GHC.IE is set, and an unknown FIS is received, the HBA shall generate an interrupt.
03	RW	0	**Set Device Bits FIS Interrupt Enable (SDBE):** When set, GHC.IE is set, and P0IS.SDBS is set, the HBA shall generate an interrupt.
02	RW	0	**DMA Setup FIS Interrupt Enable (DSE):** When set, GHC.IE is set, and P0IS.DSS is set, the HBA shall generate an interrupt.
01	RW	0	**PIO Setup FIS Interrupt Enable (PSE):** When set, GHC.IE is set, and P0IS.PSS is set, the HBA shall generate an interrupt.
00	RW	0	**Device to Host Register FIS Interrupt Enable (DHRE):** When set, GHC.IE is set, and P0IS.DHRS is set, the HBA shall generate an interrupt.

3.2.7 Offset 118h: P0CMD – Port 0 Command

Bit	Type	Reset	Description
31:28	RW	0h	**Interface Communication Control (ICC):** This is a four bit field which can be used to control reset and power states of the interface. Writes to this field shall cause actions on the interface, either primitives or an OOB sequence, and the resulting status of the interface shall be reported in the PxSSTS register. **Value** / **Definition** *Fh - 7h* *Reserved* 6h **Slumber:** This shall cause the HBA to request a transition of the interface to the slumber state. The SATA device may reject the request and the interface shall remain in its current state. *5h - 3h* *Reserved* 2h **Partial:** This shall cause the HBA to request a transition of the interface to the partial state. The SATA device may reject the request and the interface shall remain in its current state. 1h **Active:** This shall cause the HBA to request a transition of the interface into the active state. 0h **No-Op / Idle:** When software reads this value, it indicates the HBA is ready to accept a new interface control command, although the transition to the previously selected state may not yet have occurred.. When system software writes a non-reserved value other than No-Op (0h), the HBA shall perform the action and update this field back to Idle (0h). If software writes to this field to change the state to a state the link is already in (i.e. interface is in the active state and a request is made to go to the active state), the HBA shall take no action and return this field to Idle
27	RW/ RO	0	**Aggressive Slumber / Partial (ASP):** When set, and ALPE is set, the HBA shall aggressively enter the slumber state when it clears the PxCI register and the PxSACT register is cleared. When cleared, and ALPE is set, the HBA shall aggressively enter the partial state when it clears the PxCI register and the PxSACT register is cleared. See section 8.3.1.3 for details.
26	RW/ RO	0	**Aggressive Link Power Management Enable (ALPE):** When set, the HBA shall aggressively enter a lower link power state (partial or slumber) based upon the setting of the ASP bit. See section 8.3.1.3 for details.
25	RW	0	**Drive LED on ATAPI Enable (DLAE):** When set, the HBA shall drive the LED pin active for ATAPI commands (PxCLB[CHz.A] set) in addition to ATA commands. When cleared, the HBA shall only drive the LED pin active for ATA commands. See section 12.10 for details on the activity LED.
24	RW	0	**Device is ATAPI (ATAPI):** When set, the connected device is an ATAPI device. This bit is used by the HBA to control whether or not to generate the desktop LED when commands are active. See section 12.10 for details on the activity LED.
23:21	*RO*	*0*	*Reserved*
20	RO	HwInit	**Cold Port Logic Attached to Port (CPP):** When cold presence detect is supported in the platform (CAP.SCD set), this indicates whether this particular port has cold port logic attached. This bit can be used by system software to enable such features as aggressive power management, as dicsonnects can always be detected regardless of PHY state with an interlock switch. When this bit is set, it is expected that HPCP in this register is also set. The HBA takes no action on the state of this bit – it is for system software only. For example, if this bit is cleared, and the cold port logic toggles, the HBA shall still treat it as a proper cold presence detect event.

Bit	Type	Reset	Description
19	RO	HwInit	**Interlock Switch Attached to Port (ISP):** When interlock switches are supported in the platform (CAP.SIS set), this indicates whether this particular port has an interlock switch attached. This bit can be used by system software to enable such features as aggressive power management, as dicsonnects can always be detected regardless of PHY state with an interlock switch. When this bit is set, it is expected that HPCP in this register is also set. The HBA takes no action on the state of this bit – it is for system software only. For example, if this bit is cleared, and an interlock switch toggles, the HBA shall still treat it as a proper interlock switch event.
18	RO	HwInit	**Hot Plug Capable Port (HPCP):** This indicates whether the platform exposes this port to a device which can be hot plugged. SATA by definition is hot-pluggable, but not all platforms are constructed to allow the device to be removed (it may be screwed into the chassis, for example). This bit can be used by system software to indicate a feature such as “eject device” to the end-user. The HBA takes no action on the state of this bit – it is for system software only. For example, if this bit is cleared, and a hot plug event occurs, the HBA shall still treat it as a proper hot plug event.
17	RW/ RO	0	**Port Multiplier Attached (PMA):** This bit is read/write for HBAs that support a Port Multiplier (CAP.PMS = ‘1’). This bit is read-only for HBAs that do not support a port mutiplier (CAP.PMS = ‘0’). When set, a Port Multiplier is attached to the HBA for this port. When cleared, a Port Multiplier is not attached to the HBA for this port.
16	RW/ RO	0	**Port Multipler FIS Based Switching Enable (PMFSE):** This bit is read/write for HBAs that support Port Multipliers (CAP.PMS = ‘1’), and also support FIS-based switching (CAP.PMFS = ‘1’). This bit is read only ‘0’ for HBAs that do not support a port multipler (CAP.PMS = ‘0’) or support a Port Multiplier but not FIS-based switching (CAP.PMS = ‘1’, CAP.PMFS = ‘0’). When set by software, and PMA is set, the HBA knows that a port multipler is attached and FIS-based switching shall occur. When cleared, and PMA is set, only command-based switching shall occur. When cleared, and PMA is cleared, a Port Multiplier is not attached.
15	RO	0	**Command List Running (CR):** When this bit is set, the command list DMA engine for the port is running. See section the AHCI state machine in section 5.2.2 for details on when this bit is set and cleared by the HBA.
14	RO	0	**FIS Receive Running (FR):** When set, the FIS Receive DMA engine for the port is running. See section 12.2.2 for details on when this bit is set and cleared by the HBA.
13	RO	See Desc	**Interlock Switch State (ISS):** For HBAs that support interlock switches (via CAP.SIS), if an interlock switch exists on this port (via ISP in this register), this bit indicates the current state of the interlock switch. A ‘0’ indicates the switch is closed, and a ‘1’ indicates the switch is opened. For HBAs that do not support interlock switches, or if an interlock switch is not attached to this port, this bit reports ‘0’.
12:08	RO	0h	**Current Command Slot (CCS):** Indicates the current command slot the HBA is processing. This field is valid when the ST bit is set in this register, and is constantly updated by the HBA. This field can be updated as soon as the HBA recognizes an active command slot, or at some point soon after when it begins processing the command. This field is used by software to determine the current command issue location of the HBA. In queued mode, software shall not use this field, as its value does not represent the current command being executed. Software shall only use P0CI and P0SACT when running queued commands.
07:05	*RO*	*0*	*Reserved*
04	RW	0	**FIS Receive Enable (FRE):** When set, the HBA may post received FISes into the FIS receive area pointed to by PxFB (and for 64-bit HBAs, PxFBU). When cleared, received FISes are not accepted by the HBA, except for the first D2H register FIS after the initialization sequence.. System software must not set this bit until PxFB (PxFBU) have been programmed with a valid pointer to the FIS receive area, and if software wishes to move the base, this bit must first be cleared, and software must wait for the FR bit in this register to be cleared.

Bit	Type	Reset	Description
03	RW/ RO	0	**Port Selector Activate (PSA):** This bit is read/write for HBAs that support Port Selector acceleration via CAP.PSSA. This bit is read-only '0' for HBAs that do not support Port Selector acceleration. When set to '1', the HBA shall start a COMRESET sequence as described in the Port Selector specification. When cleared by software to '0', the HBA shall stop the COMRESET sequence. The HBA clears this bit when it completes the sequence. If the sequence fails, the HBA shall attempt the sequence again. This shall continue until either the sequence completes successfully, or software clears the bit.
02	RW/ RO	0/1	**Power On Device (POD):** This bit is read/write for HBAs that support cold presence detect via CAP.SCD. This bit is read only '1' for HBAs that do not support cold presence detect. When set, the HBA sets the state of a pin on the HBA to '1' so that it may be used to provide power to a cold-presence detectable port.
01	RW/ RO	0/1	**Spin-Up Device (SUD):** This bit is read/write for HBAs that support staggered spin-up via CAP.SSS. This bit is read only '1' for HBAs that do not support staggered spin-up. On an edge detect from '0' to '1', the HBA shall start a COMRESET initializatoin sequence to the device. Clearing this bit causes no action on the interface.
00	RW	0	**Start (ST):** When set, the HBA may process the command list. When cleared, the HBA may not process the command list. Whenever this bit is changed from a '0' to a '1', the HBA starts processing the command list at entry '0'. Whenever this bit is changed from a '1' to a '0', the PxCI register is cleared by the HBA upon the HBA putting the controller into an idle state.

3.2.8 Offset 120h: P0TFD – Port 0 Task File Data

This is a 32-bit register that copies specific fields of the task file when FISes are received. The FISes that contain this information are:

- D2H Register FIS
- PIO Setup FIS
- Set Device Bits FIS

Bit	Type	Reset	Description
31:16	*RO*	*0*	*Reserved*
15:08	RO	0	**Error (ERR):** Contains the latest copy of the task file error register.
07:00	RO	7Fh	**Status (STS):** Contains the latest copy of the task file status register. Fields of note in this register that affect AHCI: (see table below) The HBA shall update the entire 8-bit field, not just the bits noted above.

Bit	Field	Definition
7	BSY	Indicates the interface is busy
6:4	n/a	Not applicable
3	DRQ	Indicates a data transfer is requested
2:1	n/a	Not applicable
0	ERR	Indicates an error during the transfer.

3.2.9 Offset 124h: P0SIG – Port 0 Signature

This is a 32-bit register which contains the initial signature of an attached device when the first D2H Register FIS is received from that device. It is updated once after a reset sequence.

Bit	Type	Reset	Description
31:00	RO	FFFFFFFFh	**Signature (SIG):** Contains the signature received from a device on the first D2H Register FIS. The bit order is as follows: (see table below)

Bit	Field
31:24	LBA High Register
23:16	LBA Mid Register
15:08	LBA Low Register
07:00	Sector Count Register

3.2.10 Offset 128h: P0SSTS – Port 0 Serial ATA Status (SCR0: SStatus)

This is a 32-bit register that conveys the current state of the interface and host. The HBA updates it continuously and asynchronously.

Bit	Type	Reset	Description
31:12	*RO*	*0*	*Reserved*
11:08	RO	0	**Interface Power Management (IPM):** Indicates the current interface state: 0h Device not present or communication not established 1h Interface in active state 2h Interface in PARTIAL power management state 6h Interface in SLUMBER power management state All other values reserved
07:04	RO	0	**Current Interface Speed (SPD):** Indicates the negotiated interface communication speed. 0h Device not present or communication not established 1h Generation 1 communication rate negotiated 2h Generation 2 communication rate negotiated All other values reserved
03:00	RO	0	**Device Detection (DET):** Indicates the interface device detection and Phy state. 0h No device detected and Phy communication not established 1h Device presence detected but Phy communication not established 3h Device presence detected and Phy communication established 4h Phy in offline mode as a result of the interface being disabled or running in a BIST loopback mode All other values reserved

3.2.11 Offset 12Ch: P0SCTL – Port 0 Serial ATA Control (SCR2: SControl)

This is a 32-bit read-write register by which software controls SATA capabilities. Writes to this register result in an action being taken by the host adapter or interface. Reads from the register return the last value written to it.

Bit	Type	Reset	Description
31:16	*RO*	*0*	*Reserved*
19:16	*RO*	*0h*	*Port Multiplier Port (PMP): This field is not used by AHCI.*
15:12	*RO*	*0h*	*Select Power Management (SPM): This field is not used by AHCI*
11:08	RW	0h	**Interface Power Management Transitions Allowed (IPM):** Indicates which power states the HBA is allowed to transition to: 0h No interface restrictions 1h Transitions to the PARTIAL state disabled 2h Transitions to the SLUMBER state disabled 3h Transitions to both PARTIAL and SLUMBER states disabled All other values reserved
07:04	RW	0h	**Speed Allowed (SPD):** Indicates the highest allowable speed of the interface. 0h No speed negotiation restrictions 1h Limit speed negotiation to Generation 1 communication rate 2h Limit speed negotiation to Generation 2 communication rate All other values reserved
03:00	RW	0h	**Device Detection Initialization (DET):** Controls the HBA's device detection and interface initialization. 0h No device detection or initialization action requested 1h Perform interface communication initialization sequence to establish communication. This is functionally equivalent to a hard reset and results in the interface being reset and communications reinitialized. While this field is 1h, COMRESET is continuously transmitted on the interface. 4h Disable the Serial ATA interface and put Phy in offline mode. All other values reserved When this field is written to a 1h, the HBA shall initiate COMRESET, and start the initialization process. When the initialization is complete, this field shall remain 1h until set to another value by software. This field may only be changed to 1h or 4h when PxCMD.ST is '0'. Changing this field while the HBA is running results in undefined behavior.

3.2.12 Offset 130h: P0SERR – Port 0 Serial ATA Error (SCR1: SError)

Bit	Type	Reset	Description
31:16	RWC	0000h	**Diagnostics (DIAG)** - Contains diagnostic error information for use by diagnostic software in validating correct operation or isolating failure modes:

31:27	*Reserved*
26	**Exchanged (X):** When set to one this bit indicates a COMINIT signal was received. This bit is reflected in the interrupt register P0IS.PCS.
25	**Unrecognized FIS Type (F):** Indicates that one or more FISs were received by the Transport layer with good CRC, but had a type field that was not recognized.
24	**Transport state transition error (T):** Indicates that an error has occurred in the transition from one state to another within the Transport layer since the last time this bit was cleared.
23	**Link Sequence Error (S):** Indicates that one or more Link state machine error conditions was encountered. The Link Layer state machine defines the conditions under which the link layer detects an erroneous transition.
22	**Handshake Error (H):** Indicates that one or more R_ERR handshake response was received in response to frame transmission. Such errors may be the result of a CRC error detected by the recipient, a disparity or 8b/10b decoding error, or other error condition leading to a negative handshake on a transmitted frame.
21	**CRC Error (C):** Indicates that one or more CRC errors occurred with the Link Layer.
20	**Disparity Error (D):** Indicates that incorrect disparity was detected one or more times.
19	**10B to 8B Decode Error (B):** Indicates that one or more 10B to 8B decoding errors occurred.
18	**Comm Wake (W):** Indicates that a Comm Wake signal was detected by the Phy.
17	**Phy Internal Error (I):** Indicates that the Phy detected some internal error.
16	**PhyRdy Change (N):** Indicates that the PhyRdy signal changed state.

Bit	Type	Reset	Description
15:00	RWC	0000h	**Error (ERR):** The ERR field contains error information for use by host software in determining the appropriate response to the error condition. If one or more of bits 11:8 of this register are set, the controller shall stop the current transfer.

15:12	*Reserved*
11	**Internal Error (E):** The SATA controller failed due to a master or target abort when attempting to access system memory.
10	**Protocol Error (P):** A violation of the Serial ATA protocol was detected.
9	**Persistent Communication or Data Integrity Error (C):** A communication error that was not recovered occurred that is expected to be persistent. Persistent communications errors may arise from faulty interconnect with the device, from a device that has been removed or has failed, or a number of other causes.
8	**Transient Data Integrity Error (T):** A data integrity error occurred that was not recovered by the interface.
7:2	*Reserved*
1	**Recovered Communications Error (M):** Communications between the device and host was temporarily lost but was re-established. This can arise from a device temporarily being removed, from a temporary loss of Phy synchronization, or from other causes and may be derived from the PhyNRdy signal between the Phy and Link layers.
0	**Recovered Data Integrity Error (I):** A data integrity error occurred that was recovered by the interface through a retry operation or other recovery action.

3.2.13 Offset 134h: P0SACT – Port 0 Serial ATA Active (SCR3: SActive)

Bit	Type	Reset	Description
31:0	R/W1	0	**Device Status (DS):** System software sets this bit for SATA queuing operations prior to setting the PxCI.CI bit in the same command slot entry. This field is cleared via the Set Device Bits FIS. This field is also cleared when PxCMD.ST is written from a '1' to a '0' by software.

3.2.14 Offset 138h: P0CI – Port 0 Command Issue

Bit	Type	Reset	Description
31:0	R/W1	0	**Commands Issued (CI):** This field is set by software to indicate to the HBA that a command has been bulit in system memory for a command slot and may be sent to the device. When the HBA receives a FIS which clears the BSY, ERR, and DRQ bits for the command, it clears the corresponding bit in this register for that command slot. This field is also cleared when PxCMD.ST is written from a '1' to a '0' by software.

3.2.15 Offset 140h: P0RMCS – Raw FIS Mode Control and Status

This register is read-only ‘0’ for HBAs that do not support Raw FIS Mode (CAP.SRM = ‘0’).

Bit	Type	Reset	Description
31:16	RO	0	**Last Received FIS Length (LRFL):** This value represents the length of the last FIS received into the unknown FIS area when in raw mode. This field is recalculated upon every received FIS. The value of this field is indeterminate while a receive is in progress. The maximum value of this field is 2004h (8KB + 1 DW)
15:07	*RO*	*0*	*Reserved*
06:04	RO	0	**Transmitter Received Status (TRS):** The HBA sets this field to one of the following values, depending upon the status of the last transmitted FIS: **Bits** – **Definition** 000 – No transmissions have occurred. This is the initial state of this field whenever ERM in this register is changed from ‘0’ to ‘1’. 001 – Transmission in progress. No status returned 010 – Transmission complete. X_RDY/X_RDY collision encountered 011 – Transmission complete. R_OK returned by device. 100 – Transmission complete. R_ERR returned by device. *101 – 111* – *Reserved*
03	*RO*	*0*	*Reserved*
02	RW1	0	**Receiver Return R_ERR (RERR):** When software sets this bit to a ‘1’ the HBA shall return R_ERR to the last received FIS when in raw mode. After the HBA returns the R_ERR status to the device, the HBA shall clear this bit to ‘0’. A value of ‘0’ written by software to this bit shall have no effect.
01	RW1	0	**Receiver Return R_OK (ROK):** When software sets this bit to a ‘1’ the HBA shall return R_OK to the last received FIS when in raw mode. After the HBA returns the R_OK status to the device, the HBA shall clear this bit to ‘0’. A value of ‘0’ written by software to this bit shall have no effect.
00	RW	0	**Enable Raw FIS Mode (ERM):** When set to ‘1’, the HBA enters a special raw FIS mode. Before entering this mode, software shall allocate an 8KB + 1DW receive FIS area region and set the PxFB pointer accordingly. When this bit is set to ‘1’, the HBA shall copy all received FISes to the receive FIS area starting at offset 0. This bit is read-only for HBAs that do not support raw mode (CAP.SRM = ‘0’). See section 5.6 for details.

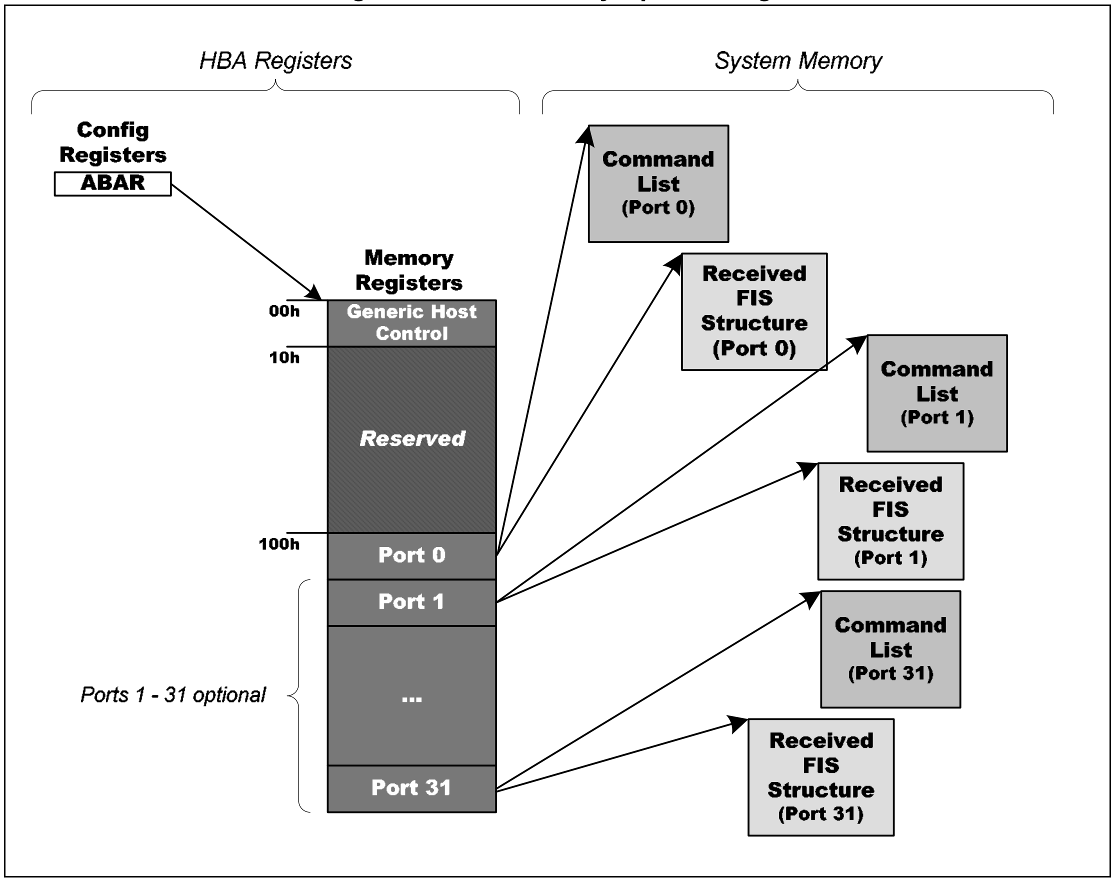
4 System Memory Structures

4.1 HBA Memory Space Usage

Most communication between software and an SATA device is through the HBA via system memory descriptors, which indicates the status of all received and sent FISes, as well as pointers for data transfers. Some additional communication is done via registers in the HBA, for each port and for global control.

A visual breakdown of the HBA memory space is shown in Figure 4.

Figure 4: HBA Memory Space Usage

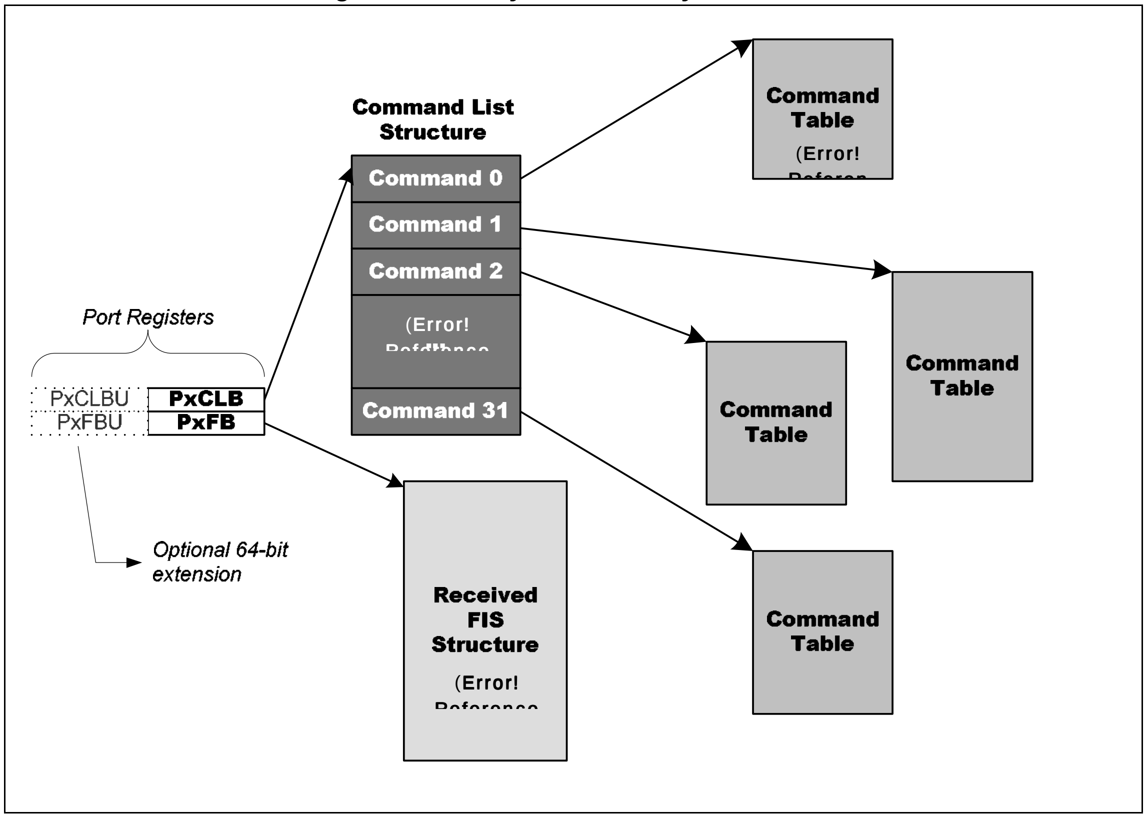


4.2 Port Memory Usage

There are two descriptors per port that are used to convey information. One is the FIS descriptor, which contains FISes received from a device, and the other is the Command List, which contains a list of 1 to 32 commands available for a port to execute.

The base for each pointer is a 64-bit value (32-bits for HBAs that do not support 64-bit addressing). An overview of the overall structure shown in Figure 5, and the following sections describe each area.

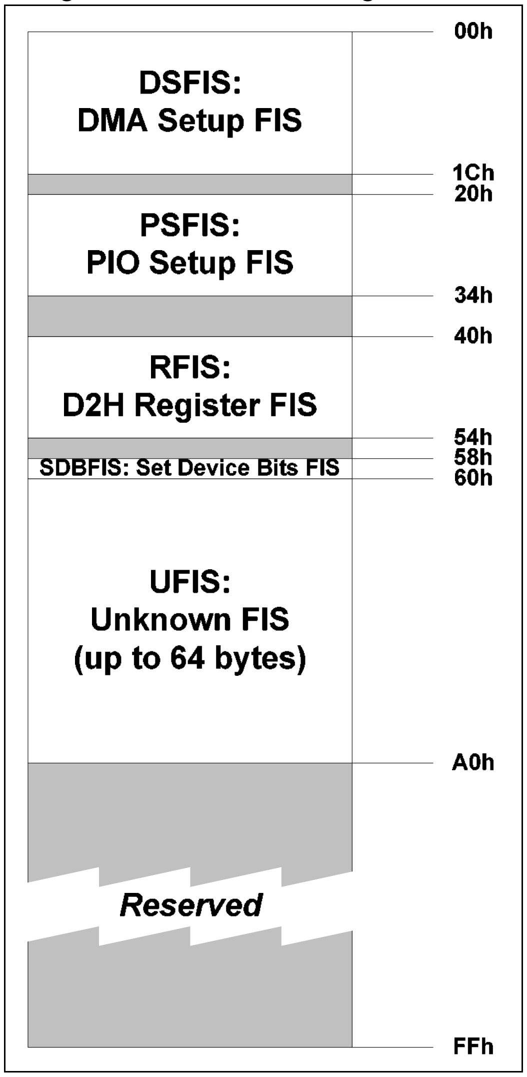
Figure 5: Port System Memory Structures



4.2.1 Received FIS Structure

The HBA uses an area of system memory to communicate information on received FISes. It is located off of PxFBU and PxFB. The structure is shown in Figure 6.

Figure 6: Received FIS Organization



When a DMA setup FIS arrives from the device, the HBA copies it to the DSFIS area of this structure, and if the 'I' bit is set, the HBA sets PxIS.DSS. Additionally, if PxIE.DSE is set, the HBA shall generate an interrupt. This allows software to see which command is currently being processed.

When a PIO setup FIS arrives from the device, the HBA copies it to the PSFIS area of this structure, and if the 'I' bit is set, sets PxIS.PSS after the data transfer. Additionally, if PxIE.PSE is set, the HBA shall generate an interrupt. This allows software to process the command and perform the appropriate actions in response to the FIS.

When a D2H Register FIS arrives from the device, the HBA copies it to the RFIS area of this structure, and if the 'I' bit is set, sets PxIS.DHRS. Additionally, if PxIE.DHRE is set, the HBA shall generate an interrupt.

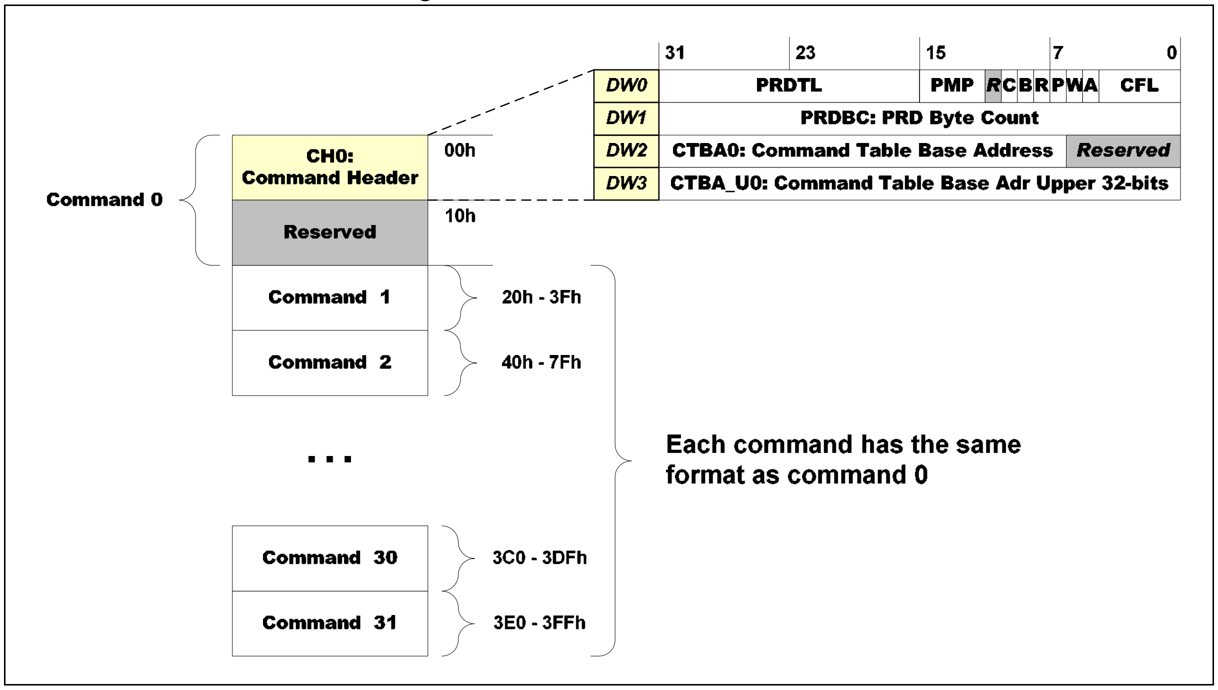
When a Set Device Bits FIS arrives from the device, the HBA copies it to the SDBFIS area of this structure, and if the 'I' bit is set, sets the PxIS.SDBS bit. Additionally, if the PxIE.SDBE bit is set, the HBA shall generate an interrupt.

When an unknown FIS arrives from the device, the HBA copies it to the UFIS area in this structure, and sets PxSERR.F, which is reflected in PxIS.US. Additionally, if the PxIE.UE bit is set, the HBA shall generate an interrupt. This allows software to process unknown FISes. A maximum of 64-bytes of an unknown FIS type may be sent to an HBA. If an unknown FIS arrives that is longer than 64-bytes, the HBA discards the extra data and performs error operations, described in section 6.1.2.. While the length of the FIS is unknown to the HBA, it is expected to be known by system software, and therefore only the valid bytes shall be processed by software.

4.2.2 Command List Structure

Figure 7 shows the command list structure. Each entry contains a command header, which is a 16-byte structure that details the direction, type, and scatter/gather pointer of the command. Further details of each field are listed below.

Figure 7: Command List Structure



The fields inside the command header are:

Figure 8: DW 0 – Description Information

Bit	Description
31:16	**Physical Region Descriptor Table Length (PRDTL):** Length of the scatter/gather descriptor table in entries, called the Physical Region Descriptor Table. Each entry is 4 DW. A '0' represents 0 entries, FFFFh represents 65,535 entries. The HBA uses this field to know when to stop fetching PRDs. If this field is '0', then no data transfer shall occur with the command.
15:12	**Port Multiplier Port (PMP):** Indicates the port number that should be used when constructing data FISes on transmit, and to check against all FISes received for this command. This value must be set to 0h if a Port Multiplier is not attached (PxCMD.PMA is '0').
11	*Reserved*
10	**Clear Busy upon R_OK (C):** When set, the HBA shall clear PxTFD.STS.BSY and PxCI.CI(z) after transmitting this FIS and receiving R_OK. When cleared, the HBA shall not clear PxTFD.STS.BSY nor PxCI.CI(z) after transmitting this FIS and receivig R_OK.
09	**BIST (B):** When '1', indicates that the command that software built is for sending a BIST FIS. The HBA shall send the FIS and enter a test mode. The tests that can be run in this mode are outside the scope of this specification.
08	**Reset (R):** When '1', indicates that the command that software built is for a device reset. The HBA must perform a SYNC escape (if necessary) to get the device into an idle state before sending the command. See section 0 for details on reset.
07	**Prefetchable (P):** This bit is only valid if the PRDTL field is non-zero. When set and PRDTL is non-zero, the HBA may prefetch PRDs in anticipation of performing a data transfer. System software must not set this bit when using queued DMA commands or if a Port Multiplier is used, and FIS-based switching is enabled.
06	**Write (W):** When set, indicates that the direction is a device write (data from system memory to device). When cleared, indicates that the direction is a device read (data from device to system memory). If this bit is set and the P bit is set, the HBA may prefetch data in anticipation of receiving a DMA Activate or PIO Setup FIS, in addition to prefetching PRDs.
05	**ATAPI (A):** When '1', indicates that a PIO setup FIS shall be sent by the device indicating a transfer for the ATAPI command. The HBA may prefetch data from CTBAz[ACMD] in anticipation of receiving the PIO Setup FIS.
04:00	**Command FIS Length (CFL):** Length of the Command FIS. A '0' represents 0 DW, '4' represents 4 DW. A length of '0' is illegal. The maximum value allowed is 10h, or 16 DW. The HBA uses this field to know the length of the FIS it shall send to the device.

Figure 9: DW 1 - Command Status

Bit	Description
31:00	**Physical Region Descriptor Byte Count (PRDBC):** Indicates the current byte count that has transferred on device writes (system memory to device) or device reads (device to system memory). For rules on when this field is updated, refer to section 5.3.1

Figure 10: DW 2 – Command Table Base Address

Bit	Description
31:07	**Command Table Descriptor Base Address (CTBA):** Indicates the 32-bit physical address of the command table, which contains the command FIS, ATAPI Command, and PRD table. This address must be aligned to a 128-byte cache line, indicated by bits 06:00 being reserved.
06:00	*Reserved*

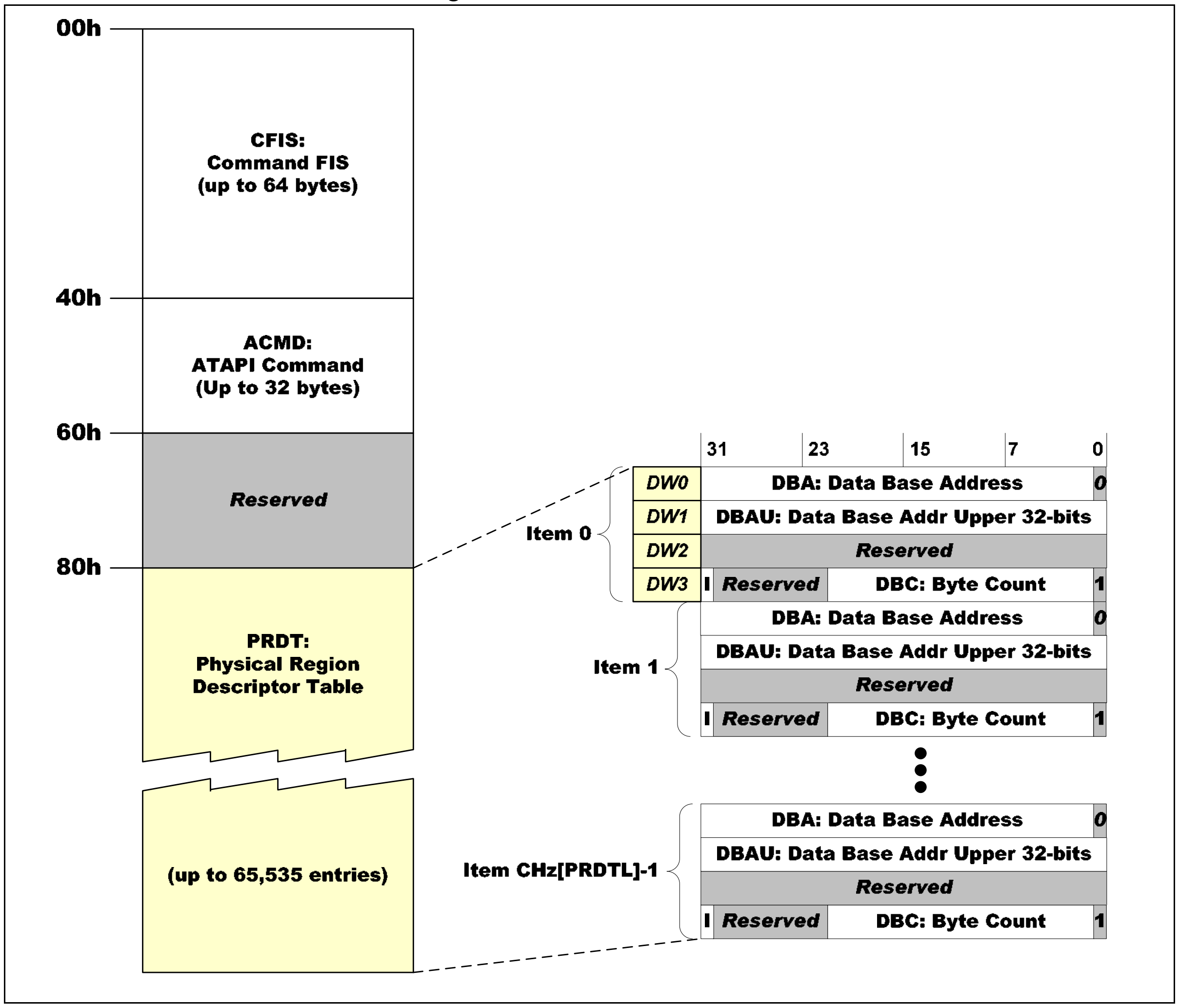
Figure 11: DW 3 – Command Table Base Address Upper

Bit	Description
31:00	**Command Table Descriptor Base Address Upper 32-bits (CTBAU):** This is the upper 32-bits of the Command Table Base. It is only valid if the HBA indicated that it can support 64-bit addressing through the S64A bit in the capabilities register, and is ignored otherwise.

4.2.3 Command Table

Each entry in the command list points to a structure called the command table.

Figure 12: Command Table



Each command contains several fields. The fields break down as follows:

4.2.3.1 Command FIS (CFIS)

This is a software constructed FIS. For data transfer operations, this is the H2D register FIS format as specified in the SATA 1.0 specification. The HBA sets PxTFD.STS.BSY, and then sends this structure to the attached port. If a Port Multiplier is attached, this field must have the Port Multiplier port number in the FIS itself – it shall not be added by the HBA.

4.2.3.2 ATAPI Command (ACMD)

This is a software constructed region of up to 32-bytes in length that contains the ATAPI command to transmit if the “A” bit is set in the command header. The length transmitted by the HBA is determined by the PIO setup FIS that is sent by the device requesting the ATAPI command.

4.2.3.3 Physical Region Descriptor Table (PRDT)

This table contains the scatter / gather list for the data transfer. It contains a list of 0 (no data to transfer) to up to 65,535 entries. A breakdown of each field in a PRD table is shown below. Item 0 refers to the first entry in the PRD table. Item "CHz[PRDTL] – 1" refers to the last entry in the table, where the length field comes from the PRDTL field in the command list entry for this command slot.

Figure 13: DW 0 – Data Base Address

Bit	Description
31:01	**Data Base Address (DBA):** Indicates the 32-bit physical address of the data block. The block must be word aligned, indicated by bit 0 being reserved.
00	*Reserved*

Figure 14: DW 1 – Data Base Address Upper

Bit	Description
31:00	**Data Base Address Upper 32-bits (DBAU):** This is the upper 32-bits of the data block physical address. It is only valid if the HBA indicated that it can support 64-bit addressing through the S64A bit in the capabilities register, and is ignored otherwise.

Figure 15: DW 2 – Reserved

Bit	Description
31:00	*Reserved*

Figure 16: DW 3 – Description Information

Bit	Description
31	**Interrupt on Completion (I):** When set, indicates that hardware should assert an interrupt when the data block for this entry has transferred, which means that no data is in the HBA hardware. Data may still be in flight to system memory (disk reads), or at the device (an R_OK or R_ERR has yet to be received). The HBA shall set the DP bit of the port status register after completing the data transfer, and if enabled, generate an interrupt.
30:22	*Reserved*
21:00	**Data Byte Count (DBC):** A '0' based value that Indicates the length, in bytes, of the data block. A maximum of length of 4MB may exist for any entry. Bit '0' of this field must always be '1' to indicate an even byte count. A value of '1' indicates 2 bytes, '3' indicates 4 bytes, etc.

5 Data Transfer Operation

5.1 Introduction

The data structures of AHCI are built assuming that each port in an HBA contains its own DMA engine, that is, each port can be executed independently of any other port. This is a natural flow for software, since each device is generally treated as a separate execution thread. It is strongly recommended that HBA implementations proceed in this fashion, as history has shown that sharing DMA engines and FIFOs within parallel ATA HBAs has been problematic.

Software presents a list of commands to the HBA for a port, which then processes them. For HBAs that have a command list depth of ‘1’, this is a single step operation, and software only presents a single command. For HBAs that support a command list, multiple commands may be posted.

Software posts new commands received by the OS to empty slots in the list, and sets the corresponding slot bit in the PxCI register. The HBA continuously looks at PxCI to determine if there are commands to transmit to the device.

The HBA processes the command list in a linear fashion, as described by the HBA state machine below.

5.2 HBA State Machine (Normative)

The state machine arcs are in priority order and are not required to be mutually exclusive. For example, if both the first arc and second arc of a state are true, the first arc is always taken. Subsequent arcs are not evaluated until the previous arc is determined to be false.

Note: In this version of the specification, raw mode is not part of the state machine.

5.2.1 Variables

hbaIssueTag	The hbaIssueTag variable contains the tag of the last command issued. On power-up or reset of the HBA port, hbaIssueTag is cleared to 0.
hbaDataTag	The hbaDataTag variable contains the tag of the command to transfer data for next. On power-up or reset of the HBA port, hbaDataTag is cleared to 0.
hbaPMP	The hbaPMP variable contains the value in the PMP field of the command table of the last command FIS transferred to the device. On power-up or reset of the HBA port, hbaPMP is cleared to 0.
hbaXferAtapi	The hbaXferAtapi variable is set to 1 when a command is issued that had the A bit set for a particular. The hbaXferAtapi variable is cleared to 0 when a Data FIS is transferred to the device that contains the ATAPI command from the command list. On power-up or reset of the HBA port, hbaXferAtapi is cleared to 0.
hbaPioESts	The hbaPioESts variable is set to the E_Status field of the PIO Setup FIS to be stored until the data for the DRQ block is transferred. On power-up or reset of the HBA port, hbaPioESts is cleared to 0.
hbaPioErr	The hbaPioErr[x] variable is set to the Error field of the PIO Setup FIS to be stored until the data for the DRQ block is transferred. On power-up or reset of the HBA port, all values in this array are cleared to 0.
hbaPioIbit	The hbaPioIbit variable is set to the Error field of the PIO Setup FIS to be stored until the data for the DRQ block is transferred. On power-up or reset of the HBA port, hbaPioIbit is cleared to 0.
hbaDmaXferCnt	The hbaDmaXferCnt variable is set to the DMA transfer count for a particular DMA transfer. The DMA transfer may consist of multiple Data FISes. The hbaDmaXferCnt variable is decremented by the size of a Data FIS on each successful reception of a Data FIS. On power-up or reset of the HBA port, hbaDmaXferCnt is cleared to 0. An hbaDmaXferCnt = ‘0’ signals that there was no DMA Setup FIS or PIO Setup FIS corresponding to the data transfer and that the transfer lengths should be constructed based on the PRD table entries only.

hbaFatal	The hbaFatal variable is set whenever the HBA has detected a fatal error. When this variable is set, no new commands shall be issued.
hbaCmdRetry	This variable is set whenever the HBA encounters an X_RDY/X_RDY collision on the interface when trying to transmit a command. It is used by the state machine to retry the last command, as opposed to fetching a new command.
hbaPrdIntr	This variable is set whenever the HBA completes a PRD in either the data transmission or data reception states. It is used to generate a PRD interrupt at the end of a successful data FIS.
hbaUpdateSig	This variable is set whenever the HBA needs to update the PxSIG register due to a hard or soft reset. It is cleared when the D2H register FIS which updates the signature is received.

5.2.2 HBA Idle States

5.2.2.1 H:Init

H:Init	The HBA performs the following actions: 1. Sets all state machine variables to '0'. 2. Resets all port specific registers as specified in section section 3		
	1. Unconditional		H:NotRunning
	NOTE: This state is entered asynchronously when GHC.HR is set to '1' from a previous value of '0'.		

The H:Init state is entered to initialize or reset a port. This state is only entered when the HBA powers up or an entire HBA reset is performed. In either case, the global IS register shall be cleared to '0' and GHC.IE shall be cleared to '0' upon transition of the first port into this state. In this state, the HBA sets all port specific state machine variables, listed in section 0 to '0' and resets all port specific registers to their reset values as specified in section 3. If cold presence detect is supported as specified in CAP.SCD, the power to each port is off by default and remains off in this state until software enables power to the port.

Arc 1: The HBA unconditionally transitions to state H:NotRunning.

5.2.2.2 H:NotRunning

H:NotRunning	HBA sets PxCMD.CR = '0'. HBA sets hbaIssueTag = 32. HBA sets z = CAP.NCS.	
1. PxCMD.POD written to '1' from a '0'	→	H:PowerOn
2. PxCMD.POD written to '0' from a '1'	→	H:PowerOff
3. PxSCTL.DET written to '4h' from any other value	→	H:Offline
4. PxSCTL.DET written to '0h' from '1h' and PxCMD.SUD = '1'	→	H:StartComm
5. PxCMD.SUD written to '1' from '0' and PxSCTL.DET = '0h'	→	H:StartComm
6. PxCMD.SUD written to '0' from '1' and PxSCTL.DET = '0h'	→	H:PhyListening
7. PxCMD.PSA written to '1' from '0' and Phy not in offline mode	→	H:PSStart
8. PxCMD.PSA written to '0' from '1' by software	→	H:PSStop
9. PxCMD.ERM written to '1' from '0'	→	H:RawEnable
10. PxCMD.ERM written to '0' from '1'	→	H:RawDisable
11. PxCMD.ST = '1' and PxSSTS.DET= 3h	→	H:Idle
12. Else	→	H:NotRunning

The H:NotRunning state is entered when the port specific DMA engines are not running. The DMA engines may not be running for numerous reasons, including because the PxCMD.ST is cleared to '0' or the communication link with the device has not been established. In this state, the HBA clears PxCMD.CR to '0', sets hbaIssueTag to 32 and sets z = CAP.NCS.

Software is only allowed to program the register bits mentioned above in this state (except PxCMD.ICC), signified by PxCMD.ST = '0' and PxCMD.CR = '0'. If it programs them in any other state, indeterminate results may occur.

Arc 1: If PxCMD.POD is written to '1' from a '0', the HBA transitions to state H:PowerOn.

Arc 2: If PxCMD.POD is written to '0' from a '1', the HBA transitions to state H:PowerOff.

Arc 3: If PxSCTL.DET is written to 4h from any other value, the HBA transitions to state H:Offline.

Arc 4: If PxSCTL.DET is written to 1h from any other value and PxCMD.SUD = '1', the HBA transitions to state H:StartComm.

Arc 5: If PxCMD.SUD is written to a '1' from a previous value of '0' and PxSCTL.DET = '0', the HBA transitions to the H:StartComm state.

Arc 6: If PxCMD.SUD is written to a '0' from a previous value of '1' and PxSCTL.DET = '0', the HBA transitions to the H:PhyListening state.

Arc 7: If PxCMD.PSA is written to '1' from '0' and the Phy is not in offline mode, the HBA transitions to state H:PSStart.

Arc 8: If PxCMD.PSA is written to '0' from '1' by software, the HBA transitions to state H:PSStop.

Arc 9: If PxCMD.ERM is written to '1' from a '0', the HBA transitions to state H:RawEnable.

Arc 10: If PxCMD.ERM is written to '0' from a '1', the HBA transitions to state H:RawDisable.

Arc 11: If PxCMD.ST = '1' and PxSSTS.DET = 3h, the HBA transitions to state H:Idle.

Arc 12: The HBA stays in state H:NotRunning as a fall-through condition.

5.2.2.3 H:Offline

H:Offline	HBA puts Phy into offline mode.	
1. Unconditional	→	H:NotRunning

The H:Offline state is entered to place the Phy for the port into offline mode. In this state, the HBA places the Phy into offline mode and the voltage level is brought to common-mode.

Arc 1: The HBA unconditionally transitions to state H:NotRunning.

5.2.2.4 H:StartBitCleared

H:StartBitCleared	HBA clears PxCI and PxSACT to 0h.		
	1. Unconditional	→	H:NotRunning
	NOTE: This state is entered asynchronously when software writes the PxCMD.ST bit to a '0' from a previous value of '1' and the HBA is not in state H:Init.		

The H:StartBitCleared state is entered when the PxCMD.ST bit is cleared to '0' from a previous value of '1'. In this state, the HBA clears PxCI to 0h and clears PxSACT to 0h.

Arc 1: The HBA unconditionally transitions to state H:NotRunning after ensuring that the DMA engines for this port are in an idle condition.

5.2.2.5 H:Idle

H:Idle	HBA sets PxCMD.CR to '1'.		
	1. PxSSTS.DET != 3h	→	H:NotRunning
	2. PxCI != 0h and hbaIssueTag = 32	→	H:SelectCmd
	3. PxCI.CI(hbaIssueTag) = '1' and hbaCmdRetry = '1' and CTBA(hbaIssueTag)[R] is set to 1 and hbaDmaXferCnt = '0'	→	CFIS:SyncEscape
	4. FIS received and PxRMCS.ERM = '1'	→	RAWR:Receive
	5. Data FIS received	→	DR:Entry
	6. Non-Data FIS received	→	NDR:Entry
	7. PxCI.CI(hbaIssueTag) = '1' and and PxRMCS.ERM = '1'	→	RAWX:Transmit
	8. PxCI.CI(hbaIssueTag) = '1' and hbaCmdRetry = '1' and hbaDmaXferCnt = '0'	→	CFIS:Xmit
	9. Else	→	H:Idle

The H:Idle state is entered when in normal operation to determine the next thing to do. In this state the HBA sets PxCMD.CR to '1'.

Arc 1: If PxSSTS.DET != 3h, the HBA transitions to state H:NotRunning.

Arc 2: If PxCI != 0h and hbaIssueTag = 32, the HBA transitions to state H:SelectCmd.

Arc 3: If PxCI.CI(hbaIssueTag) = '1' and hbaCmdRetry = '1' and CTBA(hbaIssueTag)[R] is set to '1' and hbaDmaXferCnt = '0', the HBA transitions to state CFIS:Entry.

Arc 4: If a FIS is received, and the HBA is in raw mode, the HBA transitions to state RAWR:Receive.

Arc 5: If a Data FIS is received, the HBA transitions to state DR:Entry.

Arc 6: If a FIS is received, the HBA transitions to state NDR:Entry.

Arc 7: If PxCI.CI(hbaIssueTag) = '1' and the HBA is in raw mode, the HBA transitions to state RAWX:Transmit.

Arc 8: If PxCI.CI(hbaIssueTag) = '1' and hbaCmdRetry = '1' and hbaDmaXferCnt = '0', the HBA transitions to state CFIS:Entry.

Arc 9: The HBA stays in H:Idle as a fall-through condition.

5.2.2.6 H:SelectCmd

H:SelectCmd	HBA sets z = (z + 1) mod (CAP.NCS + 1).	
1. PxCI = 0h	→	H:Idle
2. PxCI.CI(z) = ‘0’	→	H:SelectCmd
3. Else	→	H:FetchCmd

The H:SelectCmd state is entered to select the next command to issue to the device. In this state, the HBA sets z = (z + 1) mod (CAP.NCS + 1) where CAP.NCS specifies the number of command slots.

Arc 1: If PxCI = 0h, the HBA transitions to state H:Idle.

Arc 2: If PxCI.CI(z) is cleared to ‘0’, the HBA transitions to state H:SelectCmd. Note that this is written as an interative process, where each clock checks a CI value. It is shown this way to explicitly indicate the next command to be sent. An implementation may optimize the search in a single clock, so long as the correct command is selected.

Arc 3: The HBA transitions to state H:FetchCmd after a command is selected.

5.2.2.7 H:FetchCmd

H:FetchCmd	The HBA performs the following actions in the order specified: 1. HBA fetches command header from PxCLB[CHz] 2. HBA sets hbaIssueTag = z 3. HBA sets PxCMD.CCS = z 4. HBA sets hbaCmdRetry = ‘1’	
1. Unconditional	→	H:Idle

The H:FetchCmd state is entered to fetch the command header for the next command to issue from memory. In this state the HBA fetches the command header from PxCLB[CHz], sets the variable hbaIssueTag equal to z, and sets PxCMD.CCS (Current Command Slot) equal to z.

Arc 1: The HBA transitions unconditionally to state H:Idle.

5.2.2.8 H:StartComm

H:StartComm	The HBA tells link layer to start communication, which involves sending COMRESET to device. The HBA resets PxTFD to 7Fh, and sets hbaUpdateSig to ‘1’..	
1. Unconditional	→	H:NotRunning
NOTE: Hardware polling to determine if a device is present is an implementation specific detail. A polling strategy is not specified in AHCI		

The H:StartComm state is entered to start communication with the device by issuing a COMRESET. In this states, the HBA signals the link layer to start communication with the device (including issuing a COMRESET), resets PxTFD to its reset value, and resets PxSIG to its reset value.

Arc 1: The HBA unconditionally transitions to state H:NotRunning.

5.2.2.9 H:PowerOn

H:PowerOn	The HBA drives the appropriate external pin to a level which enables the FET to supply power to the device.	
1. Unconditional	→	H:NotRunning

The H:PowerOn state is used to power on a device port. In this state, the HBA drives the appropriate external pin to a level which enables the FET to supply power to the device.

Arc 1: The HBA unconditionally transitions to state H:NotRunning.

5.2.2.10 H:PowerOff

H:PowerOff	The HBA drives the appropriate external pin to a level which disables the FET from supplying power to the device.		
	1. Unconditional	→	H:NotRunning

The H:PowerOff state is used to power off a device port. In this state, the HBA drives the appropriate external pin to a level which disables the FET from supplying power to the device.

Arc 1: The HBA unconditionally transitions to state H:NotRunning.

5.2.2.11 H:PhyListening

H:SpinUp	The HBA sets the Phy into a low power mode. In this mode the Phy shall not respond to COMINIT or COMWAKE from the device. In this mode the Phy shall set PxSERR.DIAG.X to '1' if COMINIT is received.		
	1. Unconditional	→	H:NotRunning

This state is entered when PxCMD.SUD is set to '0' from a previous value of '1' and PxSCTL.DET = '0'. The HBA puts the Phy into a special mode in this state where the Phy shall not establish communication with the device but the Phy shall set PxSERR.DIAG.X to '1' if COMINIT is received.

Arc 1: The HBA unconditionally transitions to state H:NotRunning.

5.2.2.12 H:PSStart

H:PSStart	HBA starts protocol-based port selection sequence as specified in the Port Selector specification.		
	1. Unconditional	→	H:NotRunning

The H:PSStart state is used to automate the protocol-based port selection sequence for Port Selectors. In this state, the HBA starts the protocol-based port selection sequence as specified in the Port Selector specification.

Arc 1: The HBA unconditionally transitions to state H:NotRunning.

5.2.2.13 H:PSStop

H:PSStop	HBA stops protocol based port selection sequence		
	1. Unconditional	→	H:NotRunning

The H:PSStop state is used to stop the protocol-based port selection sequence for Port Selectors. In this state, the HBA stops the protocol-based port selection sequence as specified in the Port Selector specification.

Arc 1: The HBA unconditionally transitions to state H:NotRunning.

5.2.2.14 H:RawEnable

H:RawEnable	The HBA sets PxRMCS.TRS to '000'.		
	1. Unconditional	→	H:NotRunning

This state prepares the HBA for raw mode. The FIS receive structure pointed to by PxFB (and PxFBU if CAP.S64A is '1'), becomes 8KB + 2DW

Arc 1: The HBA unconditionally transitions to state H:NotRunning.

5.2.2.15 H:RawDisable

H:RawDisable			
	1. Unconditional	→	H:NotRunning

The FIS receive structure pointed to by PxFB (and PxFBU if CAP.S64A is '1') defaults to its regular size.

Arc 1: The HBA unconditionally transitions to state H:NotRunning.

5.2.3 Aggressive Power Management States

These states are entered when the HBA has no more commands to operate on and may aggressively enter a lower power state on the link.

5.2.3.1 AggrPM:Entry

AggrPM:Entry			
1.	PxCI != 0h or PxSACT != 0h	→	H:Idle
2.	PxCMD.ALPE = '1' and PxCMD.ASP = '0' and CAP.PSC = '1' and PxSCTL.IPM != '1h' and PxSCTL.IPM != '3h'	→	AggrPM:Partial
3.	PxCMD.ALPE = '1' and PxCMD.ASP = '1' and CAP.SSC = '1' and PxSCTL.IPM != '2h' and PxSCTL.IPM != '3h'	→	AggrPM:Slumber
4.	Else	→	H:Idle

In this state, the HBA determines whether it can place the link into a low power state if aggressive power management is enabled. The link can only be placed into a low power state if both the PxCI and the PxSACT registers are cleared to zero.

Arc 1: If there are commands to issue, or the queue is not empty, the HBA transitions to state H:Idle.

Arc 2: If enabled for aggressive power management, enabled for partial, the HBA is partial capable, and the port is allowed to enter the partial state, the HBA transitions to state AggrPM:Partial.

Arc 3: If enabled for aggressive power management, enabled for slumber, the HBA is slumber capable, and the port is allowed to enter the slumber state, the HBA transitions to state AggrPM:Slumber.

Arc 4: The HBA transitions to state H:Idle as a fall-through condition.

5.2.3.2 AggrPM:Partial

H:AggrPartial	HBA places link in partial state		
1.	Unconditional	→	H:Idle

In this state, the HBA attempts to place the link into a partial state as specified in the Serial ATA 1.0a specification. The HBA is not guaranteed to succeed in this operation due to a number of reasons including the current value of the PxSSTS.IPM field and whether the device responds with PMNAK$_P$ to the request.

Arc 1: The HBA transitions unconditionally to state H:Idle.

5.2.3.3 AggrPM:Slumber

H:AggrSlumber	HBA places link in slumber state		
1.	Unconditional	→	H:Idle

In this state, the HBA attempts to place the link into a slumber state as specified in the Serial ATA 1.0a specification. The HBA is not guaranteed to succeed in this operation due to a number of reasons including the current value of the PxSSTS.IPM field and whether the device responds with PMNAK$_P$ to the request.

Arc 1: The HBA transitions unconditionally to state H:Idle.

5.2.4 Non-Data FIS Receive States

5.2.4.1 NDR:Entry

NDR:Entry	Receive entire FIS into internal FIFO.		
	1. Reception error	→	ERR:Non-fatal
	2. FIS type is D2H Register FIS or PIO Setup FIS and PxTFD.STS.BSY = '0'	→	ERR:Fatal
	3. FIS Type is DMA Activate and PRD table has no remaining bytes	→	ERR:Fatal
	4. Else	→	NDR:Accept

This state is entered when the HBA has received a non-Data FIS. The HBA receives the entire FIS into its internal FIFO.

Arc 1: If the FIS reception failed, non-fatal error handling is performed.

Arc 2: If the FIS is a D2H Register or PIO Setup FIS, and PxTFD.STS.BSY = '0', a fatal error has occurred.

Arc 3: If the FIS is a DMA Activate FIS and the PRD table has no remaining bytes, a fatal error has occurred.

Arc 2: The HBA transitions to state NDR:Accept if no errors occurred.

5.2.4.2 NDR:Accept

NDR:Accept	HBA accepts the FIS with a status of R_OK.		
	1. FIS Type is D2H Register	→	RegFIS:Entry
	2. FIS Type is Set Device Bits	→	SDB:Entry
	3. FIS Type is DMA Activate	→	DX:Entry
	4. FIS Type is DMA Setup	→	DmaSet:Entry
	5. FIS Type is BIST Activate	→	BIST:Entry
	6. FIS Type is PIO Setup	→	PIO:Entry
	7. Else	→	UFIS:Entry

In this state, the FIS received has been verified to be valid. The HBA accepts the FIS with a status of R_OK.

Arc 1: If the FIS is a D2H Register FIS, the HBA transitions to state RegFIS:Entry.

Arc 2: If the FIS is a Set Device Bits FIS, the HBA transitions to state SDB:Entry.

Arc 3: If the FIS is a DMA Activate FIS, the HBA transitions to state DX:Entry.

Arc 4: If the FIS is a DMA Setup FIS, the HBA transitions to state DmaSet:Entry.

Arc 5: If the FIS is a BIST Activate FIS, the HBA transitions to state BIST:Entry.

Arc 6: If the FIS is a PIO Setup FIS, the HBA transitions to state PIO:Entry.

Arc 7: The HBA transitions to state UFIS:Entry as a fall-through since the FIS did not match all known FIS types .

5.2.5 Command Transfer States

5.2.5.1 CFIS:SyncEscape

CFIS:SyncEscape	HBA performs a SYNC escape until the interface is quiescent. HBA sets hbaUpdateSig to '1' to flag that the signature needs to be updated.		
	1. Unconditional	→	CFIS:Xmit

This state is entered when a reset FIS has been constructed to send to the device. In this state, the HBA shall perform a SYNC escape on the interface until the interface is quiescent.

Arc 1: The HBA transitions unconditionally to state CFIS:Xmit.

5.2.5.2 CFIS:Xmit

CFIS:Xmit	HBA performs the following actions in the order specified: 1. HBA sets PxTFD.STS.BSY to '1' 2. Sets hbaDataTag = hbaIssueTag, hbaPMP = CTBA(hbaIssueTag)[PMP], hbaXferAtapi = CTBA(hbaIssueTag)[A], hbaDmaXferCnt = 0 3. Fetches command FIS from CTBA(hbaIssueTag)[CFIS]. 4. Transmits FIS to device, with a length given by CTBA(hbaIssueTag)[CFL]		
	1. X_RDY/X_RDY collision on interface, delay FIS transfer	→	CFIS:Collision
	2. R_OK status received for FIS	→	CFIS:Success
	3. Else	→	CFIS:Fail

This state is entered to transmit a command FIS to the device. In this state the HBA performs the following actions in the order listed:

1. HBA sets PxTFD.STS.BSY to '1'
2. Sets hbaDataTag = hbaIssueTag, hbaPMP = CTBA(hbaIssueTag)[PMP], hbaXferAtapi = CTBA(hbaIssueTag)[A], hbaDmaXferCnt = 0
3. Fetches command FIS from CTBA(hbaIssueTag)[CFIS].
4. Transmits FIS to device, with a length given by CTBA(hbaIssueTag)[CFL]

Arc 1: If there was an X_RDY_P/X_RDY_P collision on the interface, the HBA transitions to state CFIS:Collision.

Arc 2: If R_OK status was received for the FIS, the HBA transitions to state CFIS:Success.

Arc 3: The FIS transfer failed. Recovery (including retransmission) needs to be performed.

5.2.5.3 CFIS:Success

CFIS:Success			
	1. CTBA(hbaIssueTag).B = '1'	→	BIST:Entry
	2. CTBA(hbaIssueTag).C = '1'	→	CFIS:ClearCI
	3. CTBA(hbaIssueTag).P = '1'	→	CFIS:PrefetchPRD
	4. Else	→	H:Idle

This state is entered after a command FIS is transferred successfully.

Arc 1: If the BIST (B) bit was set to '1' in the command header, the HBA transitions to state BIST:Entry.

Arc 2: If the Clear BSY upon R_OK (C) bit was set to '1' in the command header, the HBA transitions to state CFIS:ClearCI.

Arc 3: If the Prefetchable (P) bit was set to '1' in the command header, the HBA transitions to state CFIS:PrefetchPRD.

Arc 4: The HBA transitions to state H:Idle as a fall-through condition.

5.2.5.4 CFIS:Fail

CFIS:Fail	HBA sets hbaCmdRetry to ‘1’.	
1. Unconditional	→	ERR:Non-fatal

This state is entered after a command FIS transfer fails. The retry variable is set so that this command can be sent again.

Arc 1: The HBA unconditionally transitions to ERR:Non-fatal to perform error recovery.

5.2.5.5 CFIS:ClearCI

CFIS:ClearCI	HBA clears PxTFD.STS,BSY to ‘0’. HBA clears PxCI.CI(hbaIssueTag) to ‘0’. HBA sets hbaIssueTag to 32.	
1. Unconditional	→	AggrPM:Entry

This state is entered when the HBA successfully transmits a command FIS to the device and the Clear Bsy upon R_OK (C) bit is set in the command header. In this state the HBA clears PxTFD.STS.BSY to ‘0’, clears PxCI.CI(hbaIssueTag) to ‘0’ to indicate the command is complete, and sets hbaIssueTag to 32.

Arc 1: The HBA transitions unconditionally to state AggrPM:Entry.

5.2.5.6 CFIS:PrefetchPRD

CFIS:PrefetchPRD	HBA prefetches some number of PRDs, based on implementation specific algorithm and buffer space.	
1. CH(hbaIssueTag).W set	→	CFIS:PrefetchData
2. Else	→	H:Idle

This state is entered after a command FIS has been transferred and there are PRD entries that can be prefetched for the command. The HBA will prefetch some number of PRDs based on an implementation specific algorithm and on how much buffer space is available.

Arc 1: If the Write (W) bit is set to 1 in the command header, the HBA transitions to state CFIS:PrefetchData.

Arc 2: The HBA transitions to state H:Idle as a fall-through condition.

5.2.5.7 CFIS:PrefetchData

CFIS:PrefetchData	HBA prefetches some amount of data, based on implementation specific algorithm and buffer space.	
1. Unconditional	→	H:Idle

This state is entered after a command FIS has been transferred and there is data to be written to the device that can be prefetched. The HBA will prefetch some amount of data based on an implementation specific algorithm and on how much buffer space is available.

Arc 1: The HBA transitions unconditionally to state H:Idle.

5.2.5.8 CFIS:Collision

CFIS:Collision	HBA sets hbaCmdRetry to ‘1’.	
1. Unconditional	→	H:Idle

This state is entered when a command FIS could not be transferred to the device because the device asserted X_RDY$_P$ to start a transfer of its own. The HBA should behave as if the HBA never tried to issue the command FIS and should reschedule that command FIS to be issued after the FIS is received from the device. To accomplish this, the HBA sets the hbaCmdRetry variable to ‘1’.

Arc 1: The HBA transitions unconditionally to state H:Idle.

5.2.6 ATAPI Command Transfer States

The states in this portion of the state machine are responsible for transmitting the ACMD field to the device.

5.2.6.1 ATAPI:Entry

ATAPI:Entry	The HBA constructs a Data FIS with the minimum of hbaDmaXferCnt bytes or 32 bytes of data from CTBA(hbaDataTag)[ACMD]. The PMP field of the Data FIS is set to the value in PxCLB[CH(hbaDataTag)][PMP]. HBA transmits the Data FIS to the device.		
	1. Error occurred on transmission	→	ERR:Fatal
	2. Transmission successful (R_OK received)	→	ATAPI:Success

This state is entered to transmit the ATAPI command to the device. The HBA constructs a Data FIS with hbaDmaXferCnt bytes of data from CTBA(hbaDataTag)[ACMD]. The Port Multiplier Port field of the Data FIS is set to the value in PxCLB[CH(hbaDataTag)][PMP]. The HBA then proceeds to transmit the Data FIS to the device.

Arc 1: If there is an error during transmission, the HBA transitions to a fatal error state.

Arc 2: The HBA transitions to state ATAPI:Success as a fall-through condition.

5.2.6.2 ATAPI:Success

ATAPI:Success	HBA clears hbaXferAtapi to 0.		
	1. Unconditional	→	H:Idle

This state is entered after the ATAPI command is successfully transferred to the device. The HBA clears hbaXferAtapi to 0.

Arc 1: The HBA transitions unconditionally to state H:Idle.

5.2.7 D2H Register FIS Receive States

5.2.7.1 RegFIS:Entry

RegFIS:Entry	HBA performs the following actions in order: 1. Copies Register FIS to system memory at PxFB[RFIS] if PxCMD.FRE is set. 2. Updates PxTFD.ERR register with value in the Error field of the Register FIS 3. Updates PxTFD.STS register with value in the Status field of the Register FIS		
	1. PxTFD.STS.ERR = ‘1’	→	ERR:Fatal
	2. PxTFD.STS.BSY =’0’ and PxTFD.STS.DRQ =’0’	→	RegFIS:ClearCI
	3. Register FIS I bit set	→	RegFIS:SetIntr
	4. Else	→	RegFIS:UpdateSig

This state is entered after a valid D2H Register FIS arrives from the device. The purpose of this state is to update HBA registers and system memory appropriately based on the D2H Register FIS received. When in this state the HBA will perform the following actions in order:

1. Copies Register FIS to system memory at PxFB[RFIS]
2. Updates PxTFD.ERR register with value in the Error field of the Register FIS
3. Updates PxTFD.STS register with value in the Status field of the Register FIS

Arc 1: If PxTFD.STS.ERR = ‘1’, the HBA transitions to state ERR:Fatal.

Arc 2: If PxTFD.STS.BSY = ‘0’ and PxTFD.STS.DRQ = ‘0’, the HBA transitions to the RegFIS:ClearCI state.

Arc 3: If the I bit is set in the Register FIS, the HBA transitions to state RegFIS:SetIntr.

Arc 4: The HBA transitions to state RegFIS:UpdateSig as a fall-through condition.

5.2.7.2 RegFIS:ClearCI

RegFIS:ClearCI	HBA clears PxCI.CI(hbaIssueTag) to ‘0’. HBA sets hbaIssueTag to 32.		
	1. Register FIS I bit set	→	RegFIS:SetIntr
	2. Else	→	RegFIS:UpdateSig

This state is entered to clear the command issue bit in the PxCI register corresponding to the last command issued. In this state the HBA clears PxCI.CI(hbaIssueTag) to ‘0’ and sets hbaIssueTag to 32.

Arc 1: If the I bit is set in the Register FIS, the HBA transitions to state RegFIS:SetIntr.

Arc 2: The HBA transitions to state RegFIS:UpdateSig as a fall-through condition.

5.2.7.3 RegFIS:SetIntr

RegFIS:SetIntr	HBA sets PxIS.DHRS to ‘1’.		
	1. PxIE.DHRE = ‘1’	→	RegFIS:SetIS
	2. Else	→	RegFIS:UpdateSig

In this state, the HBA sets PxIS.DHRS to ‘1’.

Arc 1: If PxIE.DHRE = ‘1’, the HBA transitions to state RegFIS:SetIS.

Arc 2: The HBA transitions to state RegFIS:UpdateSig as a fall-through condition.

5.2.7.4 RegFIS:SetIS

RegFIS:SetIS	HBA sets IS.IPS(x) to ‘1’.		
	1. GHC.IE = ‘1’	→	RegFIS:GenIntr
	2. Else	→	RegFIS:UpdateSig

In this state, the HBA sets IS.IPS(x) to ‘1’.

Arc 1: If GHC.IE = ‘1’, the HBA transitions to state RegFIS:GenIntr.

Arc 2: The HBA transitions to state RegFIS:UpdateSig as a fall-through condition.

5.2.7.5 RegFIS:GenIntr

RegFIS:GenIntr	HBA generates an interrupt.		
	1. Unconditional	→	RegFIS:UpdateSig

The RegFIS:GenIntr state is entered to generate an interrupt for the port after a Register FIS interrupt has occurred. This state is only entered when GHC.IE is set. In this state, the HBA generates an interrupt. See section 0 for information on how an HBA generates an interrupt.

Arc 1: The HBA unconditionally transitions to state RegFIS:UpdateSig.

5.2.7.6 RegFIS:UpdateSig

RegFIS:UpdateSig			
	1. hbaUpdateSig = ‘1’	→	RegFIS:SetSig
	2. Else	→	AggrPM:Entry

The RegFIS:UpdateSig state is entered to check whether the PxSIG register needs to be updated.

Arc 1: If the hbaUpdateSig variable is set, the PxSIG register needs to be updated.

Arc 2: If the hbaFatal variable is ‘1’, the HBA is in a fatal error condition, and returns to Fatal:Idle

Arc 3: The HBA transitions to state AggrPM:Entry as a fall-through condition.

5.2.7.7 RegFIS:SetSig

RegFIS:SetSig	HBA updates PxSIG with appropriate fields from D2H Register FIS as outlined in 3.2.9, and clears hbaUpdateSig to ‘0’		
	1. Unconditional	→	AggrPM:Entry

The RegFIS:SetSig state updates the PxSIG register. In this state, the HBA updates the PxSIG register with the appropriate fields from the D2H Register FIS as described in section 3.2.9.

Arc 1: The HBA transitions unconditionally to state AggrPM:Entry.

5.2.8 PIO Setup Receive States

5.2.8.1 PIO:Entry

PIO:Entry	HBA performs the following actions in order: 1. Copies the PIO Setup FIS to system memory at PxFB[PSFIS] if PxCMD.FRE is set. 2. Sets hbaPioESts to E_Status field of the FIS 3. Sets hbaPioErr to Error field of the FIS 4. Sets hbaDmaXferCnt to Transfer Count field of the FIS 5. Sets PxTFD.STS register to Status field of the FIS

1. PxTFD.STS.ERR = ‘1’	→	ERR:Fatal
2. D bit in the FIS is cleared to 0 and hbaXferAtapi is set to 0	→	DX:Entry
3. D bit in the FIS is cleared to 0 and hbaXferAtapi is set to 1	→	ATAPI:Entry
4. Else	→	H:Idle

This state receives a PIO Setup FIS and updates the appropriate state based on the PIO Setup FIS. The HBA performs the following actions to finish reception of the FIS:

1. Copies the PIO Setup FIS to system memory at PxFB[PSFIS]
2. Sets hbaPioESts to E_Status field of the FIS
3. Sets hbaPioErr to Error field of the FIS
4. Sets hbaDmaXferCnt to Transfer Count field of the FIS
5. Sets PxTFD.STS register to Status field of the FIS

Arc 1: If PxTFD.STS.ERR = ‘1’, the HBA transitions to state ERR:Fatal.

Arc 2: If the direction bit in the FIS is cleared to 0 and no ATAPI command needs to be transferred, the HBA transitions to state DX:Entry to start transmission of the Data FIS.

Arc 3: If the direction bit in the FIS is cleared to 0 and an ATAPI command needs to be transferred, the HBA transitions to state ATAPI:Entry to start transmission of the ATAPI command.

Arc 4: The HBA transitions to state H:Idle to await the reception of the Data FIS from the device as a fall-through condition.

5.2.8.2 PIO:Update

PIO:Update	HBA performs the following actions: Updates PxTFD.STS with hbaPioESts Updates PxTFD.ERR with hbaPioErr Clear hbaPioXfer = ‘0’

1. PxTFD.STS.ERR = ‘1’	→	ERR:Fatal
2. PxTFD.STS.BSY = ‘0’ and PxTFD.STS.DRQ = ‘0’	→	PIO:ClearCI
3. hbaPioIbit = ‘1’	→	PIO:SetIntr
4. Else	→	H:Idle

This state updates appropriate registers after the PIO transfer has completed. In this state, the HBA updates PxTFD.STS with hbaPioESts, updates PxTFD.ERR with hbaPioErr, and clears hbaPioXfer = ‘0’.

Arc 1: If PxTFD.STS.ERR = ‘1’, the HBA transitions to state ERR:Fatal.

Arc 2: If the BSY and DRQ bits are cleared, the command is completed.

Arc 3: If the PIO Setup FIS had its ‘I’ bit set, the HBA needs to potentially generate an interrupt.

Arc 4: The HBA transitions to state H:Idle as a fall-through condition.

5.2.8.3 PIO:ClearCI

PIO:ClearCI	The HBA clears PxCI.CI(hbaIssueTag) to '0' and sets hbaIssueTag to 32.	
1. hbaPioIbit = '1'	→	PIO:SetIntr
2. Else	→	AggrPM:Entry

In this state, the HBA clears PxCI.CI(hbaIssueTag) to '0' and sets hbaIssueTag to 32. Setting hbaIssueTag to 32 enables the HBA to fetch a new command.

Arc 1: If hbaPioIbit = '1', the HBA transitions to state PIO:SetIntr.

Arc 2: The HBA transitions to state AggrPM:Entry as a fall-through condition.

5.2.8.4 PIO:SetIntr

PIO:SetIntr	Set PxIS.PSS to '1'	
1. PxIE.PSE = '1'	→	PIO:GenIntr
2. Else	→	AggrPM:Entry

In this state, the HBA sets PxIS.PSS to '1'.

Arc 1: If PxIE.PSE = '1', the HBA transitions to state PIO:GenIntr.

Arc 2: The HBA transitions to state AggrPM:Entry as a fall-through condition.

5.2.8.5 PIO:GenIntr

PIO:GenIntr	HBA generates an interrupt.	
1. Unconditional	→	AggrPM:Entry

This state is entered to generate an interrupt for the port. This state is only entered when GHC.IE is set. In this state, the HBA generates an interrupt. See section 0 for information on how an HBA generates an interrupt.

Arc 1: The HBA transitions unconditionally to state AggrPM:Entry.

5.2.9 Data Transmit States

5.2.9.1 DX:Entry

DX:Entry	The HBA constructs a Data FIS for command list entry hbaDataTag. The PMP field of the Data FIS is set to the value in PxCLB[CH(hbaDataTag)][PMP]. The HBA fetches PRDs and data from locations specified in the hbaDataTag command list entry.

1. No data to transmit and hbaPioXfer = ‘1’	→	PIO:Update
2. No data to transmit and hbaPioXfer = ‘0’	→	H:Idle
3. Else	→	DX:Start

This state starts to construct a Data FIS to transmit to the device. This state is entered after a PIO Setup, DMA Activate, or DMA Setup FIS is received. The HBA constructs a Data FIS for command list entry number hbaDataTag. The Port Multiplier Port (PMP) field of the Data FIS is filled in based on the PMP entry in the command header for the hbaDataTag entry. The HBA will fetch PRDs and data from locations specified in the command header as needed.

Arc 1: The HBA transitions directly to state H:Idle if it was at the end of the PRD table and therefore has no data to transmit.

Arc 2: The HBA transitions to state DX:XmitStart to transmit the data FIS.

5.2.9.2 DX:Transmit

DX:XmitStart	HBA transmits the Data FIS to the device. Continue to fetch PRDs and data as necessary to complete the Data FIS. As PRDs are completed, log the ‘I’ bit in the PRD into hbaPrdIntr.

1. Transmission failed	→	ERR:Fatal
2. End of PRD table reached, or maximum data FIS length of 8KB has been transferred.	→	DX:UpdateByteCount

This state transmits a Data FIS to the device. The HBA will fetch PRDs and data as necessary to complete the Data FIS.

Arc 1: If the transmission fails, a fatal error occurred.

Arc 2: The HBA transitions to state DX:UpdateByteCount at either the end of the PRD table, or after the maximum data FIS length of 8KB has been transferred.

5.2.9.3 DX:UpdateByteCount

DX:UpdateByteCount	HBA performs the following actions: 1. Increments PxCLB[CH(hbaDataTag)][PRDBC] by size of Data FIS transferred. 2. Decrements hbaDmaXferCnt by size of Data FIS transferred.

1. hbaPrdIntr = ‘1’	→	DX:PrdSetIntr
2. hbaPioXfer = ‘1’	→	PIO:Update
3. Else	→	H:Idle

This state updates transfer byte counts based on the Data FIS that was successfully transmitted. The HBA increments the byte count in PxCLB[CH(hbaDataTag)][PRDBC] by the size of the Data FIS transferred. The HBA decrements the byte count in hbaDmaXferCnt by the size of the Data FIS transferred.

Arc 1: The HBA transitions to state DX:PrdSetIntr on a successful transfer if the hbaPrdIntr variable is set.

Arc 2: If hbaPioXfer = ‘1’, the HBA transitions to state PIO:Update.

Arc 3: The HBA transitions to state H:Idle as a fall-through condition.

5.2.9.4 DX:PrdSetIntr

DX:PrdIntr	HBA sets PxIS.DPS to '1'. HBA clears hbaPrdIntr		
	1. PxIE.DPE = '1'	→	DX:PrdSetIS
	2. hbaPioXfer = '1'	→	PIO:Update
	3. Else	→	H:idle

The HBA enters this state when a PRD interrupt needs to be genreated. In this state, the HBA sets PxIS.DPS to '1', and clears the PRD interrupt variable.

Arc 1: If PxIE.DPE = '1', the HBA transitions to state DX:PrdSetIS.

Arc 2: If hbaPioXfer = '1', the HBA transitions to state PIO:Update.

Arc 3: The HBA transitions to state H:Idle as a fall-through condition.

5.2.9.5 DX:PrdSetIS

DX:PrdSetIS	HBA sets IS.IPS(x) to '1'..		
	1. GHC.IE = '1'	→	DX:PrdGenIntr
	2. hbaPioXfer = '1'	→	PIO:Update
	3. Else	→	H:Idle

In this state, the HBA sets IS.IPS(x) to '1' and generates an interrupt.

Arc 1: If GHC.IE = '1', the HBA transitions to state DX:PrdGenIntr

Arc 2: If hbaPioXfer = '1', the HBA transitions to state PIO:Update.

Arc 3: The HBA transitions to state H:Idle as a fall-through condition.

5.2.9.6 DX:PrdGenIntr

DX:GenIntr	HBA generates an interrupt.		
	1. hbaPioXfer = '1'	→	PIO:Update
	2. Else	→	H:Idle

This state is entered to generate an interrupt. See section 0 for information on how an HBA generates an interrupt

Arc 1: If hbaPioXfer = '1', the HBA transitions to state PIO:Update.

Arc 2: The HBA transitions to state H:Idle as a fall-through.

5.2.10 Data Receive States

5.2.10.1 DR:Entry

DR:Entry			
1.	PxCMD.PMA set to '1' and (PMP field does not equal hbaPMP)	→	ERR:Non-fatal
2.	Else	→	DR:Receive

This state is entered when a Data FIS is received. This state checks to make sure the Port Multiplier Port field is valid. There is no HBA action performed in this state.

Arc 1: If PxCMD.PMA is set to '1' and the PMP field in the FIS does not match hbaPMP, the HBA transitions to state ERR:Non-fatal.

Arc 2: The HBA transitions to state DR:Receive as a fall-through condition.

5.2.10.2 DR:Receive

DR:Receive HBA fetches PRD entries from the command list entry for hbaDataTag as necessary and transfers data from FIS to system memory. As PRDs complete, if their 'I' bit is set, the HBA sets hbaPrdIntr.

1.	Reception Failed	→	ERR:Fatal
2.	Reception Successful, more data arrived than could fit in PRD table	→	ERR:Non-fatal
3.	Else	→	DR:UpdateByteCount

This state is entered when a Data FIS is received. The state transfers the data in the FIS to system memory until there is no data left to transfer or the end of the PRD table is reached.

Arc 1: If the reception failed, the HBA transitions to a fatal error state.

Arc 2: If the end of the PRD table is reached but there is still data left to transfer from the FIS, the HBA transitions to a non-fatal error state.

Arc 3: At the end of the data in the FIS, the HBA transitions to state DR:UpdateByteCount as a fall-through condition.

5.2.10.3 DR:UpdateByteCount

DR:UpdateByteCount HBA accepts FIS with status of R_OK. HBA increments PxCLB[CH(hbaDataTag)][PRDBC] by number of bytes transferred to system memory. HBA decrements hbaDmaXferCnt by number of bytes transferred to system memory.

1.	hbaPrdIntr = '1'	→	DR:PrdSetIntr
2.	hbaPioXfer = '1'	→	PIO:Update
3.	Else	→	H:Idle

This state is entered after the appropriate number of bytes in a Data FIS has been transferred to system memory and the Data FIS CRC has been verified as correct. The HBA accepts the Data FIS with a status of R_OK. The HBA increments the PRD byte count, PxCLB[CH(hbaDataTag)][PRDBC], by the number of bytes transferred to system memory from the Data FIS. The HBA decrements hbaDmaXferCnt by the number of bytes transferred to system memory from the Data FIS.

Arc 1: If hbaPrdIntr = '1', the HBA transitions to state DR:PrdSetIntr.

Arc 2: If hbaPioXfer = '1', the HBA transitions to state PIO:Update.

Arc 3: The HBA transitions to state H:Idle as a fall-through condition.

5.2.10.4 DR:PrdSetIntr

DR:PrdSetIntr	HBA sets PxIS.DPS. HBA clears variable hbaPrdIntr	
1. PxIE.DPE = ‘1’	→	DR:PrdSetIS
2. hbaPioXfer = ‘1’	→	PIO:Update
3. Else	→	H:Idle

The HBA then sets the PRD interrupt status bit, PxIS.DPS.

Arc 1: If PxIE.DPE = ‘1’, the HBA transitions to state DR:PrdSetIS.

Arc 2: If hbaPioXfer = ‘1’, the HBA transitions to state PIO:Update.

Arc 3: The HBA transitions to H:Idle as a fall-through condition.

5.2.10.5 DR:PrdSetIS

DR:PrdSetIS	HBA sets IS.IPS(x) to ‘1’..	
1. GHC.IE = ‘1’	→	DR:PrdGenIntr
2. hbaPioXfer = ‘1’	→	PIO:Update
3. Else	→	H:Idle

In this state, the HBA sets IS.IPS(x) to ‘1’ and generates an interrupt.

Arc 1: If GHC.IE = ‘1’, the HBA transitions to state DR:PrdGenIntr

Arc 2: If hbaPioXfer = ‘1’, the HBA transitions to state PIO:Update.

Arc 3: The HBA transitions to state H:Idle as a fall-through condition.

5.2.10.6 DR:PrdGenIntr

DR:PrdGenIntr	HBA generates an interrupt.	
1. hbaPioXfer = ‘1’	→	PIO:Update
2. Else	→	H:Idle

This state is entered to generate an interrupt. See section 0 for information on how an HBA generates an interrupt

Arc 1: If hbaPioXfer = ‘1’, the HBA transitions to state PIO:Update.

Arc 2: The HBA transitions to state H:Idle as a fall-through condition.

5.2.11 DMA Setup Receive States

5.2.11.1 DmaSet:Entry

DmaSet:Entry	The HBA performs the following actions: 1. HBA stores TAG field from the DMA Setup FIS in hbaDataTag 2. HBA stores TransferCount field from the DMA Setup FIS in hbaDmaXferCnt 3. HBA writes the FIS to system memory at PxFB[DSFIS] if PxCMD.FRE is set.		
	1. I bit in the FIS is set to 1	→	DmaSet:SetIntr
	2. Else	→	DmaSet:AutoActivate

This state is entered when a correctly formed DMA Setup FIS is received. In this state the HBA performs the following actions:

1. HBA stores TAG field from the DMA Setup FIS in hbaDataTag
2. HBA stores the Transfer Count field from the DMA Setup FIS in hbaDmaXferCnt
3. HBA writes the FIS to system memory at PxFB[DSFIS].

Arc 1: If the I bit is set to 1 in the FIS, the HBA transitions to state DmaSet:SetIntr.

Arc 2: The HBA transitions to state DmaSet:AutoActivate as a fall-through condition.

5.2.11.2 DmaSet:SetIntr

DmaSet:SetIntr	HBA sets PxIS.DSS to '1'.		
	1. PxIE.DSE = '1'	→	DmaSet:SetIS
	2. Else	→	DmaSet:AutoActivate

This state is entered when a DMA Setup FIS is received that has the I bit set. In this state the HBA sets PxIS.DSS to '1'.

Arc 1: If PxIE.DSE is set to 1, the HBA transitions to state DmaSet:SetIS.

Arc 2: The HBA transitions to state DmaSet:AutoActivate as a fall-through condition.

5.2.11.3 DmaSet:SetIS

DmaSet:SetIntr	HBA sets IS.IPS(x) to '1'.		
	1. GHC.IE = '1'	→	DmaSet:GenIntr
	2. Else	→	DmaSet:AutoActivate

In this state, the HBA sets IS.IPS(x) to '1'.

Arc 1: If GHC.IE = '1', the HBA transitions to state DmaSet:GenIntr.

Arc 2: The HBA transitions to state DmaSet:AutoActivate as a fall-through condition.

5.2.11.4 DmaSet:GenIntr

DmaSet:GenIntr	HBA generates an interrupt.		
	1. Unconditional	→	DmaSet:AutoActivate

This state is entered to generate an interrupt. See section 0 for information on how an HBA generates an interrupt

Arc 1: The HBA transitions to state DmaSet:AutoActivate.

5.2.11.5 DmaSet:AutoActivate

DmaSet:AutoActivate		
1. CH(hbaDataTag).W is set to 1 and A bit in the FIS is set to 1	→	DX:Entry
2. Else	→	H:Idle

This state is entered to determine whether a Data FIS should immediately be transmitted to the device.

Arc 1:	If the D bit in the FIS is cleared to 0 and the A bit in the FIS is set to 1, the HBA transitions to state DX:Entry.
Arc 2:	The HBA transitions to state H:Idle as a fall-through condition.

5.2.12 Set Device Bits States

5.2.12.1 SDB:Entry

SDB:Entry	HBA performs the following actions: 1. Copies Set Device Bits FIS to system memory at offset PxFB[SDFIS] if PxCMD.FRE is set. 2. Updates PxTFD.STS with non-reserved bits in Status field of FIS 3. Updates PxTFD.ERR with Error field of FIS 4. Clears bits in PxSACT that have corresponding bits set in the SActive field of the SDB FIS. 5. Clears hbaDmaXferCnt to '0'

1. PxTFD.STS.ERR = '1'	→	ERR:Fatal
2. I bit is set in the SDB FIS	→	SDB:SetIntr
3. Else	→	AggrPM:Entry

This state receives a Set Device Bits FIS and updates the appropriate state based on the Set Device Bits FIS. The HBA performs the following actions to finish reception of the FIS:

1. Copies Set Device Bits FIS to system memory at offset PxFB[SDFIS]
2. Updates PxTFD.STS with non-reserved bits in Status field of FIS
3. Updates PxTFD.ERR with Error field of FIS
4. Clears bits in PxSACT that have corresponding bits set in the SActive field of the SDB FIS.
5. Clears hbaDmaXferCnt to '0'

Arc 1: If PxTFD.STS.ERR = '1', the HBA transitions to state ERR:Fatal.

Arc 2: If the interrupt (I) bit is set in the FIS, the HBA transitions to SDB:SetIntr.

Arc 3: The HBA transitions to AggrPM:Entry as a fall-through condition.

5.2.12.2 SDB:SetIntr

SDB:SetIntr	HBA sets PxS.DSS to '1'.

1. PxIE.DSE = '1'	→	SDB:SetIS
2. Else	→	AggrPM:Entry

This state is entered when a Set Device Bits FIS with the interrupt bit set was received. The HBA sets PxIS.DSS to '1' in this state.

Arc 1: If PxIE.DSE = '1', the HBA transitions to SDB:SetIS.

Arc 2: The HBA transitions to AggrPM:Entry as a fall-through condition.

5.2.12.3 SDB:SetIS

SDB:SetIS	HBA sets IS.IPS[x] bit.

1. GHC.IE bit set	→	SDB:GenIntr
2. Else	→	AggrPM:Entry

This state is entered to set status bits in the global interrupt status register. Prior to entering this state, the HBA has set the appropriate PxIS bit based on the interrupt cause.

Arc 1: If GHC.IE = '1', the HBA transitions to state SDB:GenIntr.

Arc 2: The HBA transitions to state AggrPM:Entry as a fall-through condition.

5.2.12.4 SDB:GenIntr

SDB:GenIntr	HBA generates an interrupt.

1. Unconditional	→	AggrPM:Entry

This state is entered to generate an interrupt for the port. This state is only entered when GHC.IE is set. In this state, the HBA generates an interrupt. See section 0 for information on how an HBA generates an interrupt.

Arc 1: The HBA transitions unconditionally to state AggrPM:Entry.

5.2.13 Unknown FIS Receive States

5.2.13.1 UFIS:Entry

UFIS:Entry	HBA performs the following actions in order: 1. Copies the Unknown FIS to system memory at PxFB[UFIS] if PxCMD.FRE is set. 2. Sets PxSERR.F to '1' 3. Sets PxIS.US to '1'

1. FIS is longer than 64-bytes	→	ERR:Fatal
2. PxIE.UE is set to 1	→	UFIS:SetIS
3. Else	→	H:Idle

This state is entered when an Unknown FIS is received and the CRC for that FIS is correct. The HBA accepts the unknown FIS with a status of R_OK. The HBA then copies up to 64 bytes of data from the unknown FIS to the UFIS part of the FIS receive area in system memory. Then the HBA sets PxSERR.F to '1' to indicate that an unknown FIS was received.

Arc 1: If the unknown FIS is longer than 64-bytes, a fatal error has occurred.

Arc 2: If PxIE.UE = '1', the HBA transitions to state UFIS:SetIS.

Arc 3: The HBA transitions to state H:Idle as a fall-through condition.

5.2.13.2 UFIS:SetIS

UFIS:SetIS	HBA sets IS.IPS[x] bit.

1. GHC.IE bit set		UFIS:GenIntr
2. Else	→	H:Idle

This state is entered to set status bits in the global interrupt status register.

Arc 1: If GHC.IE = '1', the HBA transitions to state UFIS:GenIntr.

Arc 2: The HBA transitions to state H:Idle as a fall-through condition.

5.2.13.3 UFIS:GenIntr

UFIS:GenIntr	HBA generates an interrupt.

1. Unconditional	→	H:Idle

This state is entered to generate an interrupt for the port. This state is only entered when GHC.IE is set. In this state, the HBA generates an interrupt. See section 0 for information on how an HBA generates an interrupt.

Arc 1: The HBA transitions unconditionally to state H:Idle.

5.2.14 BIST States

5.2.14.1 BIST:Entry

BIST:Entry	HBA sets PxSSTS.DET to 4h to put Phy in offline mode.

1. Unconditional	→	BIST:Entry

This state is entered when a BIST Activate FIS is sent by the HBA to the device or received from the device. In this state, the HBA sets PxSSTS.DET to 4h to put the Phy in offline mode.

Software exits the HBA from this state by clearing PxCMD.ST.

Arc 1: The HBA transitions unconditionally to state BIST:Entry.

5.2.15 Raw FIS Transfer States

5.2.15.1 RAWX:Transmit

RAWX:Transmit	HBA performs the following actions in the order specified: 5. HBA sets PxTFD.STS.BSY to '1' 6. HBA sets PxRMCS.TRS to '001'. 7. The HBA fetches PRDs and data from locations specified in the hbaDataTag command list entry, and transmits them raw to the device to satisfy the length of the FIS. If a PRD entry has the 'I' bit set, the HBA sets the hbaPrdIntr variable.

4. R_OK status received for FIS	→	RAWX:Success
5. X_RDY/X_RDY collision	→	RAWX:Collision
6. Else (R_ERR returned)	→	RAWX:Fail

This state is entered to transmit a command FIS to the device. In this state the HBA performs the following actions in the order listed:

5. Sets PxTFD.STS.BSY to '1'
6. Sets PxRMCS.TRS to '001'
7. Fetches raw FIS from PRD table.
8. Transmits FIS to device until PRD table ends.

Arc 1: If R_OK status was received for the FIS, the HBA transitions to state RAWX:Success.

Arc 1: If there was an X_RDY$_P$/X_RDY$_P$ collision on the interface, the HBA transitions to state RAWX:Collision.

Arc 3: The FIS transfer failed. The HBA unconditionally transitions to RAWX:Fail

5.2.15.2 RAWX:Success

RAWX:Success	HBA sets PxRMCS.TRS to '011'

5. hbaPrdIntr = '1'	→	RAWX:PrdSetIntr
6. Else	→	RAWX:ClearCI

This state is entered after a raw FIS is transferred successfully.

Arc 1: If the hbaPrdIntr variable is set, the HBA transitions to RAWX:PrdSetIntr.

Arc 2: The HBA transitions to state RAWX:ClearCI as a fall-through condition.

5.2.15.3 RAWX:Collision

RAWX:Collision	HBA sets PxRMCS.TRS to '010'

2. hbaPrdIntr = '1'	→	RAWX:PrdSetIntr
3. Else	→	RAWX:ClearCI

This state is entered if an X_RDY/X_RDY collision occurs on a raw FIS transmission.

Arc 1: If the hbaPrdIntr variable is set, the HBA transitions to RAWX:PrdSetIntr.

Arc 2: The HBA transitions to state RAWX:ClearCI as a fall-through condition.

5.2.15.4 RAWX:Fail

RAWX:Fail	HBA sets PxRMCS.TRS to '100'

1. hbaPrdIntr = '1'	→	RAWX:PrdSetIntr
2. Else	→	RAWX:ClearCI

This state is entered after a raw FIS transfer fails.

Arc 1: If the hbaPrdIntr variable is set, the HBA transitions to RAWX:PrdSetIntr.

Arc 2: The HBA transitions to state RAWX:ClearCI as a fall-through condition.

5.2.15.5 RAWX:PrdSetIntr

RAWX:PrdIntr	HBA sets PxIS.DPS to '1'. HBA clears hbaPrdIntr		
	4. PxIE.DPE = '1'	→	RAWX:PrdSetIS
	5. Else	→	RAWX:ClearCI

The HBA enters this state when a PRD interrupt needs to be generated. In this state, the HBA sets PxIS.DPS to '1', and clears the PRD interrupt variable.

Arc 1: If PxIE.DPE = '1', the HBA transitions to state RAWX:PrdSetIS.

Arc 2: The HBA transitions to state RAWX:ClearCI as a fall-through condition.

5.2.15.6 RAWX:PrdSetIS

RAWX:PrdSetIS	HBA sets IS.IPS(x) to '1'..		
	4. GHC.IE = '1'	→	RAWX:PrdGenIntr
	5. Else	→	RAWX:ClearCI

In this state, the HBA sets IS.IPS(x) to '1' and generates an interrupt.

Arc 1: If GHC.IE = '1', the HBA transitions to state DX:PrdGenIntr

Arc 2: The HBA transitions to state RAWX:ClearCI as a fall-through condition.

5.2.15.7 RAWX:PrdGenIntr

RAWX:GenIntr	HBA generates an interrupt.		
	3. Unconditional	→	RAWX:ClearCI

This state is entered to generate an interrupt. See section 0 for information on how an HBA generates an interrupt

Arc 1: The HBA unconditionally transitions to state RAWX:ClearCI.

5.2.15.8 RAWX:ClearCI

RAWX:ClearCI	HBA clears PxTFD.STS,BSY to '0'. HBA clears PxCI.CI(hbaIssueTag) to '0'. HBA sets hbaIssueTag to 32.		
	2. Unconditional	→	AggrPM:Entry

This state is entered when the HBA either successfully or unsuccessfully transmits a raw FIS to the device. In this state the HBA clears PxTFD.STS.BSY to '0', clears PxCI.CI(hbaIssueTag) to '0' to indicate the command is complete, and sets hbaIssueTag to 32.

Arc 1: The HBA transitions unconditionally to state AggrPM:Entry.

5.2.16 Raw FIS Receive States

5.2.16.1 RAWR:Receive

RAWR:Receive	HBA writes data from the received FIS to PxFB (and PxFBU if CAP.S64A is set).	
4. Reception Completed (Data FIS Ends)	→	RAWR:UpdateByteCount
5. Else	→	RAWR:Receive

This state is entered when a Raw FIS is received. The state transfers the data in the FIS to system memory at the FIS receive area.

Arc 1: When the device stops transmitting the raw FIS, the HBA transitions to the RAWR:UpdateByteCount state.

Arc 2: The HBA remains in the RAWR:Receive state as long as a FIS is being transmitted by the device..

5.2.16.2 RAWR:UpdateByteCount

RAWR:UpdateByteCount	HBA updates PxRMCS.LFRL with the number of bytes transferred to system memory. HBA writes the DWord of CRC received from the device into system memory.	
4. Unconditional	→	RAWR:SetIntr

This state is entered after a raw FIS has been transferred to system memory The HBA updates the byte count in PxRMCS.LFRL with the number of bytes transferred to system memory from the raw FIS. It also writes the CRC sent from the device so that software can perform a CRC check on the received data.

Arc 2: The HBA unconditionally transitions to state RAWR:SetIntr.

5.2.16.3 RAWR: SetIntr

RAWR:SetIntr	HBA sets PxIS.US.	
4. PxIE.UE = ‘1’	→	RAWR:SetIS
5. Else	→	RAWR:WaitResponse

The HBA then sets the interrupt status bit, PxIS.US.

Arc 1: If PxIE.UE = ‘1’, the HBA transitions to state DR:SetIS.

Arc 3: The HBA transitions to RAWR:WaitResponse as a fall-through condition.

5.2.16.4 RAWR:SetIS

RAWR:SetIS	HBA sets IS.IPS(x) to ‘1’..	
4. GHC.IE = ‘1’	→	RAWR:GenIntr
5. Else	→	RAWR:WaitResponse

In this state, the HBA sets IS.IPS(x) to ‘1’ and generates an interrupt.

Arc 1: If GHC.IE = ‘1’, the HBA transitions to state DR:GenIntr

Arc 3: The HBA transitions to state RAWR:WaitResponse as a fall-through condition.

5.2.16.5 RAWR:GenIntr

RAWR:GenIntr	HBA generates an interrupt.	
3. Else	→	RAWR:WaitResponse

This state is entered to generate an interrupt. See section 0 for information on how an HBA generates an interrupt

Arc 1: The HBA unconditionally transitions to state RAWR:WaitResponse.

5.2.16.6 RAWR:WaitResponse

RAWR:PrdGenIntr	HBA awaits for software to program a response		
	1. Software writes PxRMCS.ROK to ‘1’ from ‘0’	→	RAWR:ROK
	2. Software writes PxRMCS.RERR to ‘1’ from ‘0’	→	RAWR:RERR
	3. Else	→	RAWR:WaitResponse

This state is entered to await a response from software to the previously arrived FIS.

Arc 1: The HBA transitions to RAWR:ROK if software asks for an R_OK to be returned.
Arc 2: The HBA transitions to RAWR:RERR if software asks for an R_ERR to be returned.
Arc 3: The HBA remains in state RAWR:WaitResponse as a fall-through condition.

5.2.16.7 RAWR:ROK

RAWR:ROK	HBA returns R_OK to the device. HBA clears PxRMCS.ROK.		
	1. Unconditional	→	H:Idle

This state is entered to return R_OK to the device that generated the raw FIS.

Arc 1: The HBA unconditionally transitions to state H:Idle

5.2.16.8 RAWR:RERR

RAWR:RERR	HBA returns R_ERR to the device. HBA clears PxRMCS.RERR.		
	1. Unconditional	→	H:Idle

This state is entered to return R_ERR to the device that generated the raw FIS.

Arc 1: The HBA unconditionally transitions to state H:Idle

5.2.17 Error States

5.2.17.1 ERR:Fatal

This state is entered when the HBA has encountered a condition it cannot recover from. Appropriate error bits shall be set by the HBA, and interrupts may optionally be generated. See section 6 for fatal conditions. When a fatal condition occurs, the HBA requires PxCMD.ST to be cleared.

5.2.17.2 ERR:Non-Fatal

This state is entered when the HBA has encountered an error it can recover from. Appropriate error bits shall be set by the HBA, and interrupts may optionally be generated. See section 6 for fatal conditions. When a non-fatal condition occurs, the HBA shall return to H:Idle after processing the error.

5.3 HBA Rules (Normative)

5.3.1 PRD Byte Count Updates

Each command has a PRD byte count field that is initialized to ‘0’ by software prior to starting a transfer, and is updated by hardware as transfers occur. The purpose of this field is to let software know how many bytes transferred for a given operation, to determine if the transfer occurred properly.

It also gives status information while a transfer is occurring indicating how much data has been transferred in a currently executing command. This information may be useful to the system for application performance, as data being read from a device may be handed back to the operating system and application prior to the full transfer completing.

The field shall be updated at the end of each Data FIS, when the CRC for the Data FIS is determined to be correct. This guarantees that the data that was just transferred arrived correctly at the destination. For device writes (system memory to device), this is when R_OK is returned by the device. For device reads (device to system memory), this may be done when R_OK is scheduled for return, and received data has been pushed toward system memory. (The data may still be in the HBA as long as the byte count update pushes that data ahead of it).

The HBA must not update the byte count prior to the end of the Data FIS, because this data could have an error. If the field was updated prior to receiving a valid CRC, software cannot properly determine what data was transferred correctly.

If an overflow occurs, the byte count shall not be updated. The byte count that remains represents the last Data FIS that arrived prior to the overflow.

Implementation Note: There may be some variance in when the byte count is updated relative to internal FIFO sizes. For example, a sequence of very small Data FISes could theoretically be sent by a device, and these may not get drained to system memory right away. An HBA is allowed to “collapse” the Data FISes, and perform the PRD byte count update at a convenient point.

5.3.2 PRD Interrupt

When a PRD is exhausted, the HBA may be told to generate an interrupt via the ‘I’ bit in the PRD entry. Note, though, that a PRD is not considered exhausted until the Data FIS is complete. For example, if the Data FIS is 8 KB, and this is covered by 3 PRD entries, the data is not considered valid at the end of the first or second PRD, since CRC has not yet been checked, even though the data has been copied to memory or the device.

Therefore, if the ‘I’ bit is set in the PRD entry, the HBA must hold onto it internally and not set PxIS.DPS until the Data FIS is complete and CRC is correct. Once correct, PxIS.DPS can be set, and if PxIE.IE and GHC.IE are set, the HBA shall generate an interrupt.

5.4 System Software Rules (Normative)

5.4.1 Basic Steps when Building a Command

When software builds a command for the HBA to execute, it first finds an empty command slot by reading the PxCI register for the port. After a free slot (slot z), is found:

1. Software builds a command FIS in system memory at location PxCLB[CH(z)]:CFIS with the command type.
2. If it is an ATAPI command, the ACMD field shall be filled in with the ATAPI command
3. Software builds a command header at PxCLB[CH(z)] with:
   a. PRDTL containing the number of entries in the PRD table
   b. CFL set to the length of the command in the CFIS area
   c. A bit set if it is an ATAPI command
   d. W (Write) bit set if data is going to the device
   e. P (Prefetch) bit optionally set (see rules in section 5.4.2)
   f. If a Port Multiplier is attached, the PMP field set to the correct Port Multiplier port.
4. If it is a queued command, software shall first set PxSACT.DS(z).
5. Software shall set PxCI.CI(z) to indicate to the HBA that a command is active.

5.4.2 Setting CH(z).P

When software builds commands for the HBA to execute, it may optionally set CH(z).P to enable prefetching of PRDs and data. The HBA is not required to prefetch, but may use this bit to do so. To avoid potential problems where the HBA may prefetch items it cannot use, software must obey the following rules:

- Software shall not set CH(z).P when building queued ATA commands. The S-ATA device may run the commands in a different order than what is sent.
- Software shall not set CH(z).P when building commands to several devices behind a Port Multiplier when FIS based switching is enabled. FISes may be received from a Port Multiplier for a different device than what was just sent by the HBA.

If software does not obey these rules, indeterminate HBA behavior may result.

5.4.3 Processing Completed Commands

Software processes the interrupt generated by the device for command completion. In the interrupt service routine, software checks IS.IPS to determine the ports that have an interrupt pending.

For each port that has an interrupt pending:

1. Software determines the cause of the interrupt by reading the PxIS register. It is possible for multiple bits to be set
2. Software clears appropriate bits in the PxIS register corresponding to the cause of the interrupt.
3. Software clears the interrupt bit in IS.IPS corresponding to the port.
4. Software reads the PxCI register, and compares the current value to a previously read value. It completes with success any commands whose bit has been cleared since the last value was read.
5. If there were errors, noted either in the PxIS register or PxTFD.STS.ERR, software performs error recovery actions (see section 6.2.2).

5.5 Transfer Examples (Informative)

In all the following examples, it is assumed that PxCMD.ST has been set, and the HBA is only waiting for a command to be placed in the command list. If PxCMD.ST is not set, software must do so at some point. Additionally, software must ensure that the device has not changed connection status since the last command was added by checking PxIS.PCS.

At any point, the link may be in a partial or slumber state. The HBA must check the link prior to sending any FISes, and bring the link to an active state if necessary (see section 8.3.1.4).

5.5.1 Macro States

In the following examples, the HBA traverses a series of states in the HBA state machine when performing data transfers. To simplify the text in the examples, state sequences that are repeated are abbreviated here in what are called macro-states. Each of these macro-states assumes no errors were encountered.

Macro State	HBA State Machine States
Exam:Fetch	H:Idle → H:SelectCmd → H:FetchCmd → H:Idle
Exam:Transmit	H:Idle → CFIS:Xmit → CFIS:Success → H:Idle
Exam:AcceptNonData	H:Idle → NDR:Entry → NDR:Accept
Exam:DMAReceive	DR:Entry → DR:Receive → DR:UpdateByteCount → H:Idle
Exam:DMATransmit	DX:Entry → DX:Transmit → DX:UpdateByteCount → H:Idle
Exam:PIOTransmit	PIO:Entry → DX:Entry → DX:Transmit → DX:UpdateByteCount → PIO:Update → PIO:ClearCI → PIO:SetIntr → PIO:GenIntr → ChkAggrPM:Entry → H:Idle
Exam:PIOReceive	DR:Entry → DR:Receive → DR:UpdateByteCount → PIO:Update → PIO:ClearCI → PIO:SetIntr → PIO:GenIntr → ChkAggrPM:Entry → H:Idle
Exam:D2HIntr	RegFIS:Entry → RegFIS:ClearCI → RegFIS:SetIntr → RegFIS:SetIS → RegFIS:GenIntr → RegFIS:UpdateSig → ChkAggrPM:Entry → H:Idle
Exam:D2HNoIntr	RegFIS:Entry → RegFIS:ClearCI → RegFIS:UpdateSig → H:ChkAggrPM → H:Idle
Exam:DMASetup	DmaSet:Entry → DmaSet:SetIntr → DmaSet:SetIS → DmaSet:GenIntr → DmaSet:AutoActivate → H:Idle

5.5.2 DMA Data Transfers

5.5.2.1 ATA DMA Write

Software builds a command as described in section 5.4.1. The command shall have a PRD table, is not ATAPI, and is not queued. It is a DMA write (data to device), therefore CH(z).W (Write) shall be set, and CH(z).P (Prefetch) may optionally be set per the rules described in section 5.4.2.

The HBA shall transfer the command to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. If CH(z).P was set, the HBA shall execute the following states after CFIS:Success in the **Exam:Transmit** macro state before returning to idle: CFIS:PrefetchPRD → CFIS:PrefetchData → H:Idle

As this was a DMA write command, the response from the device shall be a DMA Activate FIS. When this arrives, the HBA shall accept the FIS by traversing the **Exam:AcceptNonData** macro state, and shall send a Data FIS by traversing the **Exam:DMATransmit** macro state.

If the Data FIS did not satisfy the transfer count, another DMA Activate FIS shall be sent from the device, and the HBA shall traverse the **Exam:AcceptNonData** and **Exam:DMATransmit** macro states again to send another Data FIS. This process continues until the transfer count is satisfied. When the data FIS that completes the transfer count finishes, the next FIS from the device shall be a D2H Register FIS.

The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set, the HBA shall traverse the **Exam:D2HIntr** macro state. If the 'I' bit was not set, the HBA shall traverse the **Exam:D2HNoIntr** state.

If this was the last command, and the HBA was enabled for aggressive power management, the HBA shall first request the link to be placed in either the partial or slumber power management after the AggrPM:Entry state.

5.5.2.2 ATAPI Packet DMA Write

Software builds a command as described in section 5.4.1. The command shall have a PRD table, is ATAPI, and is not queued. It is a DMA read (data to device), therefore CH(z).W (Write) shall be set, and CH(z).P (Prefetch) may optionally be set per the rules described in section 5.4.2.

The HBA shall transfer the command to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. If CH(z).P was set, the HBA shall execute the following states after CFIS:Success in the **Exam:Transmit** macro state before returning to idle: CFIS:PrefetchPRD → CFIS:PrefetchData → H:Idle

Since this was an ATAPI command, the next FIS from the device shall be a PIO Setup FIS. The HBA traverses the **Exam:AcceptNonData** macro state to accept this FIS, and then traverses the following states to transfer the ACMD field to the device: PIO:Entry → ATAPI:Entry → ATAPI:Success → H:Idle

As this command results in a DMA write, the response from the device shall be a DMA Activate FIS. When this arrives, the HBA shall accept the FIS by traversing the **Exam:AcceptNonData** macro state, and shall send a Data FIS by traversing the **Exam:DMATransmit** macro state.

If the Data FIS did not satisfy the transfer count, another DMA Activate FIS shall be sent from the device, and the HBA shall traverse the **Exam:AcceptNonData** and **Exam:DMATransmit** macro states again to send another Data FIS. This process continues until the transfer count is satisfied. When the data FIS that completes the transfer count finishes, the next FIS from the device shall be a D2H Register FIS.

The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set, the HBA shall traverse the **Exam:D2HIntr** macro state. If the 'I' bit was not set, the HBA shall traverse the **Exam:D2HNoIntr** state.

If this was the last command, and the HBA was enabled for aggressive power management, the HBA shall first request the link to be placed in either the partial or slumber power management after the AggrPM:Entry state.

5.5.2.3 ATA DMA Read

Software builds a command as described in section 5.4.1. The command shall have a PRD table, is not ATAPI, and is not queued. It is a DMA write (data to memory), therefore CH(z).W (Write) shall be cleared, and CH(z).P (Prefetch) may optionally be set per the rules described in section 5.4.2.

The HBA shall transfer the command to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. If CH(z).P was set, the HBA shall execute the following states after CFIS:Success in the **Exam:Transmit** macro state before returning to idle: CFIS:PrefetchPRD → H:Idle

As this was a DMA read command, the response from the device shall be a Data FIS. When this arrives, the HBA shall traverse the **Exam:DMARead** macro state.

If the Data FIS did not satisfy the transfer count, another Data FIS shall be sent from the device, and the HBA shall traverse the **Exam:DMARead** macro states again. This process continues until the transfer count is satisfied. When the data FIS that completes the transfer count finishes, the next FIS from the device shall be a D2H Register FIS.

The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set, the HBA shall traverse the **Exam:D2HIntr** macro state. If the 'I' bit was not set, the HBA shall traverse the **Exam:D2HNoIntr** state.

If this was the last command, and the HBA was enabled for aggressive power management, the HBA shall first request the link to be placed in either the partial or slumber power management after the AggrPM:Entry state.

5.5.2.4 ATAPI Packet DMA Read

Software builds a command as described in section 5.4.1. The command shall have a PRD table, is not ATAPI, and is not queued. It is a DMA write (data to memory), therefore CH(z).W (Write) shall be cleared, and CH(z).P (Prefetch) may optionally be set per the rules described in section 5.4.2.

The HBA shall transfer the command to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. If CH(z).P was set, the HBA shall execute the following states after CFIS:Success in the **Exam:Transmit** macro state before returning to idle: CFIS:PrefetchPRD → H:Idle

Since this was an ATAPI command, the next FIS from the device shall be a PIO Setup FIS. The HBA traverses the **Exam:AcceptNonData** macro state to accept this FIS, and then traverses the following states to transfer the ACMD field to the device: PIO:Entry → ATAPI:Entry → ATAPI:Success → H:Idle

As this command results in a DMA read, the response from the device shall be a Data FIS. When this arrives, the HBA shall traverse the **Exam:DMARead** macro state.

If the Data FIS did not satisfy the transfer count, another Data FIS shall be sent from the device, and the HBA shall traverse the **Exam:DMARead** macro states again. This process continues until the transfer count is satisfied. When the data FIS that completes the transfer count finishes, the next FIS from the device shall be a D2H Register FIS.

The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set, the HBA shall traverse the **Exam:D2HIntr** macro state. If the 'I' bit was not set, the HBA shall traverse the **Exam:D2HNoIntr** state.

If this was the last command, and the HBA was enabled for aggressive power management, the HBA shall first request the link to be placed in either the partial or slumber power management after the AggrPM:Entry state.

5.5.3 PIO Data Transfers

The following sections describe data transfers to and from a device using PIO as the command type. AHCI only allows PIO transfers to contain a single DRQ block. While this is sub-optimal, AHCI has chosen not to optimize for PIO data transfers. One of the overriding reasons for this is that PIO commands do not provide adequate error coverage. However, some commands may only be available as a PIO data transfer, (such as Identify Device), therefore AHCI provides rudimentary mechanisms to accomplish this.

If software attempts to set up a multi-DRQ block PIO transfer, undefined results shall occur in the HBA. .

From the HBA's point of view, PIO data transfers look like a DMA transfer. A command table is set up, and the data is mastered from or to system memory by the HBA. There is no data port for the CPU read or write within the controller.

5.5.3.1 ATA PIO Write

Software builds a command as described in section 5.4.1. The command shall have a PRD table, is not ATAPI, and is not queued. It is a PIO Write (data to device), therefore CH(z).W (Write) shall be set, and CH(z).P (Prefetch) may optionally be set per the rules described in section 5.4.2.

The HBA shall transfer the command to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. If CH(z).P was set, the HBA shall execute the following states after CFIS:Success in the **Exam:Transmit** macro state before returning to idle: CFIS:PrefetchPRD → CFIS:PrefetchData → H:Idle

As this was a PIO write command, the response from the device shall be a PIO Setup FIS. When this arrives, the HBA shall accept the FIS by traversing the **Exam:AcceptNonData** macro state. It shall then traverse the **Exam:PIOTransmit** macro state to send a data FIS.

Since this was PIO write command, the device shall next send a D2H Register FIS. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set, the HBA shall traverse the **Exam:D2HIntr** macro state. If the 'I' bit was not set, the HBA shall traverse the **Exam:D2HNoIntr** state.

If this was the last command, and the HBA was enabled for aggressive power management, the HBA shall first request the link to be placed in either the partial or slumber power management after the AggrPM:Entry state.

5.5.3.2 ATAPI Packet PIO Write

Software builds a command as described in section 5.4.1. The command shall have a PRD table, is not ATAPI, and is not queued. It is a PIO Write (data to device), therefore CH(z).W (Write) shall be set, and CH(z).P (Prefetch) may optionally be set per the rules described in section 5.4.2.

The HBA shall transfer the command to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. If CH(z).P was set, the HBA shall execute the following states after CFIS:Success in the **Exam:Transmit** macro state before returning to idle: CFIS:PrefetchPRD → CFIS:PrefetchData → H:Idle

Since this was an ATAPI command, the next FIS from the device shall be a PIO Setup FIS. The HBA traverses the **Exam:AcceptNonData** macro state to accept this FIS, and then traverses the following states to transfer the ACMD field to the device: PIO:Entry → ATAPI:Entry → ATAPI:Success → H:Idle

As this was a PIO Write command, the response from the device shall be a PIO Setup FIS. When this arrives, the HBA shall accept the FIS by traversing the **Exam:AcceptNonData** macro state. It shall then traverse the **Exam:PIOTransmit** macro state to send a data FIS.

Since this was an ATAPI command, the device shall send a D2H Register FIS. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set, the HBA shall traverse the **Exam:D2HIntr** macro state. If the 'I' bit was not set, the HBA shall traverse the **Exam:D2HNoIntr** state.

If this was the last command, and the HBA was enabled for aggressive power management, the HBA shall first request the link to be placed in either the partial or slumber power management after the AggrPM:Entry state.

5.5.3.3 ATA PIO Read

Software builds a command as described in section 5.4.1. The command shall have a PRD table, is not ATAPI, and is not queued. It is a PIO Read (data to memory), therefore CH(z).W (Write) shall be cleared, and CH(z).P (Prefetch) may optionally be set per the rules described in section 5.4.2.

The HBA shall transfer the command to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. If CH(z).P was set, the HBA shall execute the following states after CFIS:Success in the **Exam:Transmit** macro state before returning to idle: CFIS:PrefetchPRD → H:Idle

As this was a PIO read command, the response from the device shall be a PIO Setup FIS. When this arrives, the HBA shall accept the FIS by traversing the **Exam:AcceptNonData** macro state. It shall then traverse the states PIO:Entry → H:Idle, and await a data FIS from the device.

When the data FIS arrives, the HBA traverses the **Exam:PIORead** macro state.

If this was the last command, and the HBA was enabled for aggressive power management, the HBA shall first request the link to be placed in either the partial or slumber power management after the AggrPM:Entry state.

5.5.3.4 ATAPI Packet PIO Read

Software builds a command as described in section 5.4.1. The command shall have a PRD table, is ATAPI, and is not queued. It is a PIO Read (data to memory), therefore CH(z).W (Write) shall be cleared, and CH(z).P (Prefetch) may optionally be set per the rules described in section 5.4.2.

The HBA shall transfer the command to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. If CH(z).P was set, the HBA shall execute the following states after CFIS:Success in the **Exam:Transmit** macro state before returning to idle: CFIS:PrefetchPRD → H:Idle

Since this was an ATAPI command, the next FIS from the device shall be a PIO Setup FIS. The HBA traverses the **Exam:AcceptNonData** macro state to accept this FIS, and then traverses the following states to transfer the ACMD field to the device: PIO:Entry → ATAPI:Entry → ATAPI:Success → H:Idle

As this was a PIO read command, the response from the device shall be a PIO Setup FIS. When this arrives, the HBA shall accept the FIS by traversing the **Exam:AcceptNonData** macro state. It shall then traverse the states PIO:Entry → H:Idle, and await a data FIS from the device.

When the data FIS arrives, the HBA traverses the **Exam:PIORead** macro state.

Since this was an ATAPI command, the device shall next send a D2H Register FIS. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set, the HBA shall traverse the Exam:D2HIntr macro state. If the 'I' bit was not set, the HBA shall traverse the **Exam:D2HNoIntr** state.

If this was the last command, and the HBA was enabled for aggressive power management, the HBA shall first request the link to be placed in either the partial or slumber power management after the AggrPM:Entry state.

5.5.4 HBA Assisted Queued DMA Transfers

5.5.4.1 Introduction

Legacy ATA queued DMA, using the DRQ, SERV, and REL bits, is not supported by AHCI. Queued operations are sent to an SATA device using the “ReadFPDMAQueued” and “WriteFPDMAQueued” commands.

To allow a simple mechanism for the HBA to map command list slots to queue entries, software must match the tag number it uses to the slot it is placing the command in. For example, if a queued command is placed in slot 5, the tag for that command must be 5.

System software must determine the maximum tag allowed by the device and the HBA and it must use the lower bound of the two. For example, if the HBA has 8 entries in its command list, and the SATA device only has 4, only tags 0 – 3 in the device may be used, and only command list entries 0 – 3 may be used in the HBA.

Data transfers are activated with the flow described below via the DMA Setup FIS, and command completion is performed via the Set Device Bits FIS.

5.5.4.2 Example

In the following example, the following occurs:

- System software places 4 commands in system memory:
  - Slot 0 contains a queued DMA read
  - Slot 2 contains a queued DMA write
  - Slot 5 contains a queued DMA read
  - Slot 8 contains a queued DMA write
- The HBA fetches the first 3 commands, transfers them to the device, and receives a successful completion.
- Before the HBA can send the 4th command to the device, the device sends a DMA Setup FIS to transfer data for the command in slot 2.
- A data transfer occurs to slot 2.
- The HBA transfers slot 8 to the device.
- A data transfer occurs to slot 5.
- The device sends an SDB FIS to clear slots 2 and 5.
- A data transfer occurs to slot 8.
- The device sends an SDB FIS to clear slot 8.
- A data transfer occurs to slot 0.
- The device sends an SDB FIS to clear slot 0.

Other items to note in this data flow:

- In both queued DMA read commands, the auto-activate bit is not set in the FIS.
- Every SDB FIS received has its ‘I’ bit set.

Following is a text description for the flow.

System Software Places 4 Commands in System Memory

Software builds a command into slot 0 as described in section 5.4.1. The command shall have a PRD table, is not ATAPI, and is of type ReadFPDMAQueued. CH(z).W (Write) shall be set, and CH(z).P (Prefetch) shall be cleared.

Software builds a command into slot 2 as described in section 5.4.1. The command shall have a PRD table, is not ATAPI, and is of type WriteFPDMAQueued. Both CH(z).W (Write) and CH(z).P (Prefetch) shall be cleared.

Software builds a command into slot 5 as described in section 5.4.1. The command shall have a PRD table, is not ATAPI, and is of type WriteFPDMAQueued. Both CH(z).W (Write) and CH(z).P (Prefetch) shall be cleared.

Software builds a command into slot 8 as described in section 5.4.1. The command shall have a PRD table, is not ATAPI, and is of type ReadFPDMAQueued. CH(z).W (Write) shall be set, and CH(z).P (Prefetch) shall be cleared.

At this point, the PxCI and PxSACT register values are "00000125h" (bits 0, 2, 5, and 8 set).

The HBA transmits the First 3 Commands to the Device

The HBA shall transfer the command from slot 0 to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. As this was a queued DMA command, the next FIS from the device shall be a D2H Register FIS.

The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set, the HBA shall traverse the Exam:D2HIntr macro state. If the 'I' bit was not set, the HBA shall traverse the **Exam:D2HNoIntr** state. PxCI is now equal to "00000124h".

The HBA shall transfer the command from slot 2 to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. As this was a queued DMA command, the next FIS from the device shall be a D2H Register FIS.

The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set, the HBA shall traverse the Exam:D2HIntr macro state. If the 'I' bit was not set, the HBA shall traverse the **Exam:D2HNoIntr** state. PxCI is now equal to "00000120h".

The HBA shall transfer the command from slot 5 to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. As this was a queued DMA command, the next FIS from the device shall be a D2H Register FIS.

The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set, the HBA shall traverse the Exam:D2HIntr macro state. If the 'I' bit was not set, the HBA shall traverse the **Exam:D2HNoIntr** state. PxCI is now equal to "00000100h".

DMA Setup FIS Arrives for slot 2

The HBA is now in an idle state, but before it can fetch a new command, a short FIS arrives from the device. This FIS is the DMA Setup FIS. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state, and then traverses the **Exam:DMASetup** macro state to process the DMA Setup FIS. The tag indicated in the FIS was for slot 2.

Data Transfer for slot 2

As this was a DMA write command, and the auto-activate bit was not set in the FIS, the next FIS from the device shall be a DMA Activate FIS. When this arrives, the HBA shall accept the FIS by traversing the **Exam:AcceptNonData** macro state, and shall send a Data FIS by traversing the **Exam:DMATransmit** macro state.

If the Data FIS did not satisfy the transfer count, another DMA Activate FIS shall be sent from the device, and the HBA shall traverse the **Exam:AcceptNonData** and **Exam:DMATransmit** macro states again to send another Data FIS. This process continues until the transfer count is satisfied. The state machine is now in H:Idle

HBA transfers slot 8 to the device

Since PxCI is not all 0h, and no other FIS is coming in from the device, the HBA shall transfer the command from slot 8 to the device traversing the macro states **Exam:Fetch** and **Exam:Transmit**. As this was a queued DMA command, the next FIS from the device shall be a D2H Register FIS.

The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state. If the D2H Register FIS had the 'I' bit set, the HBA shall traverse the **Exam:D2HIntr** macro state. If the 'I' bit was not set, the HBA shall traverse the **Exam:D2HNoIntr** state. PxCI is now equal to "00000000h".

Data transfer to slot 5

A short FIS arrives from the device of type DMA Setup FIS. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state, and then traverses the **Exam:DMASetup** macro state to process the DMA Setup FIS. The tag indicated in the FIS was for slot 5.

As this was a DMA read command, the next FIS from the device shall be a Data FIS. When this arrives, the HBA shall traverse the **Exam:DMARead** macro state.

If the Data FIS did not satisfy the transfer count, another Data FIS shall be sent from the device, and the HBA shall traverse the **Exam:DMARead** macro states again. This process continues until the transfer count is satisfied. The HBA state machine is now in H:Idle.

Device sends SDB FIS to clear slots 2 and 5

At this point, the device sends an SDB FIS to indicate slots 2 and 5 are complete. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state, and then traverses the SDB:Entry, and, since the received FIS had its I bit set, the SDB:SetIntr → SDB:SetIS → SDB:GenIntr states, and returns to H:Idle. The PxSACT register is now equal to “00001001h”.

Data transfer occurs to slot 8

A short FIS arrives from the device of type DMA Setup FIS. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state, and then traverses the **Exam:DMASetup** macro state to process the DMA Setup FIS. The tag indicated in the FIS was for slot 8.

As this was a DMA write command, and the auto-activate bit was not set in the FIS, the next FIS from the device shall be a DMA Activate FIS. When this arrives, the HBA shall accept the FIS by traversing the **Exam:AcceptNonData** macro state, and shall send a Data FIS by traversing the **Exam:DMATransmit** macro state.

If the Data FIS did not satisfy the transfer count, another DMA Activate FIS shall be sent from the device, and the HBA shall traverse the **Exam:AcceptNonData** and **Exam:DMATransmit** macro states again to send another Data FIS. This process continues until the transfer count is satisfied. The HBA state machine is now in H:Idle.

Device sends SDB FIS to clear slot 8

At this point, the device sends an SDB FIS to indicate slot 8 is complete. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state, and then traverses the SDB:Entry, and, since the received FIS had its I bit set, the SDB:SetIntr → SDB:SetIS → SDB:GenIntr states, and returns to H:Idle. The PxSACT register is now equal to “00000001h”.

Data transfer occurs to slot 0

A short FIS arrives from the device of type DMA Setup FIS. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state, and then traverses the **Exam:DMASetup** macro state to process the DMA Setup FIS. The tag indicated in the FIS was for slot 2.

As this was a DMA read command, the next FIS from the device shall be a Data FIS. When this arrives, the HBA shall traverse the **Exam:DMARead** macro state.

If the Data FIS did not satisfy the transfer count, another Data FIS shall be sent from the device, and the HBA shall traverse the **Exam:DMARead** macro states again. This process continues until the transfer count is satisfied. The HBA state machine is now in H:Idle.

Device sends SDB FIS to clear slot 0

At this point, the device sends an SDB FIS to indicate slot 0 is complete. The HBA shall accept this FIS by traversing the **Exam:AcceptNonData** macro state, and then traverses the SDB:Entry, and, since the received FIS had its I bit set, the SDB:SetIntr → SDB:SetIS → SDB:GenIntr states, and returns to H:Idle. The PxSACT register is now equal to “00000000h”.

Since the PxCI and PxSACT registers are now both equal to “00000000h”, if the HBA was enabled for aggressive power management, the HBA shall first request the link to be placed in either the partial or slumber power management after the AggrPM:Entry state.

5.6 Raw FIS Mode

For HBAs that support Raw FIS mode via CAP.SRM, an additional register is added for each port, called PxRMCS. This register allows raw mode to be enabled, controls whether R_ERR or R_OK is returned on received FISes, and indicates the length of the last received FIS.

5.6.1 Enabling Raw Mode

To enable raw mode of operation, software must first bring the port to an idle condition by clearing PxCMD.ST to ‘0’ and waiting for PxCMD.CR to return to ‘0’. After PxCMD.CR is ‘0’, software shall write the PxFB register with the pointer to the raw FIS receive area that is 8KB + 2DW in length. When this is accomplished, software then sets PxRMCS.ERM.

When raw mode is enabled, the FIS receive area changes from the structure defined in section 4.2.1, to a flat structure that is 8K + 2DW in length.

5.6.2 FIS Transmission in Raw Mode

In raw mode, only command slot 0 shall be used. Software must only place commands into slot 0, and shall only set the PxCI bit for slot 0. Additionally, it may not queue any commands by setting PxSACT bits. If software sets any other PxCI or sets any bits in PxSACT, indeterminate results may occur.

The FIS that software wishes the HBA to transmit is located in the PRD table for command slot 0. The entire FIS is placed into the PRD table. Software must set the PRD interrupt bit for the last PRD entry, so that an interrupt may be generated upon the end of transmission.

The HBA shall transmit the FIS from the PRD table to the device when in raw mode. It shall not transmit any data from the CFIS area. The HBA shall only monitor the status of CH0[R] to determine whether or not to send a SYNC Escape before transmitting the FIS. All other attribute bits shall be ignored.

Upon the start of transmission, the HBA shall clear PxRMCS.TRS to ‘001’, to indicate no status has been returned from the device. If an X_RDY/X_RDY collision occurs, the FIS transmission failed, and PxRMCS.TRS is updated to ‘010’ and PxCI[0] is cleared to ‘0’ to indicate the transmission is complete. Upon the end of transmission, the HBA awaits the reception of either R_OK or R_ERR, and updates PxRMCS.TRS. When a transmission fails due to an X_RDY/X_RDY collision, software may retry the FIS transmission by setting the PxCI[0] bit again without clearing and resetting the PxCMD.ST bit. When a transmission fails due to any other reason, software must follow the normal fatal error recovery procedures.

Since software must set the PRD interrupt for this PRD, the HBA sets PxIS.DPS upon the reception of R_OK or R_ERR, and if PxIE.DPE and GHC.IE are set, generates an interrupt. The HBA then clears PxCI[0] to indicate that raw FIS transmission is complete.

5.6.3 FIS Reception in Raw Mode

All received FISes are copied into the receive FIS area in raw mode.

When in raw mode, received FISes are copied into the FIS receive area, starting at offset 0h from the PxFBU:PxFB register. All FISes are copied into the structure, and the length in PxRMCS.LFRL is updated, regardless of whether or not an error is detected. (In normal mode, this would cause the FIS to be rejected). Additionally, the CRC generated by the device is copied into the FIS receive area. Note: the data in the FIS receive area will therefore be one dWord longer than PxRMCS.LFRL. Software can then calculate its own CRC from PxRMCS.LFRL, and compare it to the last dWord sent from the device.

Upon the end of FIS reception, the HBA will hold the interface in wait states, and set PxIS.US. If enabled through PxIS.UE and GHC.IE, the HBA will generate an interrupt. Software completes the transaction by writing to either PxRMCS.ROK (return R_OK), or PxRMCS.RERR (return R_ERR).

When a second FIS arrives, the HBA overwrites the last received FIS. It is software's responsibility to save any necessary contents from the receive FIS area before returning R_OK or R_ERR status to the device for that FIS.

5.6.4 Exiting Raw Mode

To exit raw mode, software clears PxCMD.ST and awaits PxCMD.CR to clear to '0'. Software then clears PxRMCS.ERM to '0'.

6 Error Reporting and Recovery

All errors within an HBA occur within ports. There are no errors that apply to the entire host controller. There are several sources of errors that could occur during a transfer. Examples of errors are:

- System Memory – Bad system memory pointers cause data fetches and stores to be lost
- Interface / Device - such as CRC problems, illegal state machine transitions, etc.

6.1 Error Types

6.1.1 System Memory Errors

System memory errors such as target abort, master abort, and parity may cause the host to stop processing the currently running command. These are serious errors that cannot be recovered from without software intervention (section 6.2.2).

A master/target abort error occurs when system software has given a pointer to the HBA that does not exist in physical memory. When this occurs, the HBA aborts the transfer (if necessary) as described in section 6.2.1. When this is complete, the HBA sets PxIS.HBFS. If PxIE.HBFE and GHC.IE are set, the HBA shall also generate an interrupt. .

A data error (such as uncorrectable ECC, CRC, or parity), may or may not be transient. If the error occurred on a fetch of a CFIS, PRD entry or command list, the HBA shall stop. If the error occurred on a data FIS or the ACMD field, the HBA is allowed to stop, but may also continue. When a data error occurs, the HBA aborts the transfer (if necessary) as described in section 6.2.1. When this is complete, the HBA sets PxIS.HBDS. If PxIE.HBDE and GHC.IE are set, the HBA shall also generate an interrupt.

If the HBA continue after a data error on a data or ACMD field, it shall poison the CRC of the data FIS it transfers to the device.

6.1.2 Interface Errors

Interface errors are errors that occur due to electrical issues on the interface, or protocol miscommunication between the device and HBA. Depending on the type of error, different bits in the PxSERR register are set. When these bits are set, either PxIS.IFS (fatal) or PxIS.INFS (non-fatal) shall be set, and if enabled, the HBA shall generate an interrupt.

Bits in the PxSERR register that cause PxIS.IFS/PxIS.INFS to be set are:

- In the PxSERR.ERR field, the M, P, and E bits
- In the PxSERR.DIAG field, the D, C, and H bits

Examples of these types of errors are below, with the corresponding PxSERR bit that is set.

The only difference between PxIS.IFS and PxIS.INFS being set is the type of FIS that is being transmitted/received when the error occurs. If the error occurred during a non-Data FIS, the FIS must be retransmitted, so the error is non-fatal and PxIS.INFS is set. If the error occurred during a Data FIS, the transfer shall stop, so the error is fatal and PxIS.IFS is set.

In the case of a non-data FIS error, between seeing a non-data FIS fail and the attempt to re-transmit, the HBA may receive other FISes from the device (this will most likely happen when running queued commands). When this occurs, the HBA must accept the FIS, perform the correct actions, and then retry its failed FIS.

If the HBA was transmitting a Data FIS it returns to an idle state. The HBA shall retransmit a non-Data FIS continuously after a failure until either the transfer succeeds or system software stops the controller by clearing PxCMD.ST.

- **Received Disparity Error / Illegal Character (K28.3):** When a disparity error is encountered, the HBA assumes the character is correct and resets the disparity counter. No error bits are set.
- **Received Disparity Error / Illegal Character (D):** When this occurs, the HBA returns R_ERR at the end of the FIS. It sets PxSERR.DIAG.D
- **PHYReady Dropping Unexpectedly:** When this occurs, the HBA returns to idle, and sets PxSERR.ERR.M.

- **Calculated Different CRC than Received:** When this occurs, the HBA returns R_ERR and returns to idle. It sets PxSERR.DIAG.C
- **Incorrect FIS or Illegal FIS Length for Corresponding FIS Type Received:** When this occurs, the HBA returns R_ERR at end of the FIS and returns to idle. It sets PxSERR.ERR.P. This can only be done for supported FIS types. An unsupported FIS is not considered an illegal FIS, unless the length received is more than 64 bytes. If an unsupported FIS arrives, it is posted and the HBA continues normal operation.
- **Internal Buffer Overflow:** This occurs when the HBA sends a HOLD, but a HOLDA was not received quickly enough by the HBA, and the HBA's internal data FIFOs overflow. The HBA returns R_ERR at the end of the FIS. It sets PxSERR.ERR.E.
- **HBA Receives R_ERR:** If the HBA receives an R_ERR to a FIS it was transmitting, it sets PxSERR.DIAG.H.
- **FIS received from a device, where the status register is to be updated, PxTFD.STS.BSY is cleared, and the BSY bit is set:** When this occurs, the HBA returns R_OK, sets PxSERR.ERR.P, and sets PxIS.IFS. Even though this is a non-data FIS, this type of error is considered fatal, as the device should not be sending status information with PxTFD.STS.BSY set.
- **'A' bit set in CH[z], PIO Setup FIS arrives with byte count greater than 32 bytes:** The ACMD field in AHCI is only 32 bytes, and ATAPI commands are currently only 12 or 16 bytes. If the PIO Setup FIS which transfers the ATAPI command is larger than this, R_ERR is returned, and PxSERR.ERR.P is set.

It is system software's responsibility to check the PxSERR register periodically to determine if the interface is operating cleanly, and take appropriate actions (such as going down to GEN1 speed if operating at a higher speed or notifying the user) when interface errors occur.

Note that when an error such as CRC or in illegal length occurs, the HBA cannot trust the incoming FIS to be correct. For example, the HBA may have thought the FIS was a D2H Register FIS, but if the CRC is incorrect, it could be because the type specified in the FIS is incorrect.

To address this problem, the HBA shall not update its internal registers, nor update the FIS received area, until it determines that a non-Data FIS it received was valid. If the HBA believes the FIS to be a Data FIS, however, it may copy it to memory at the appropriate PRD location. This is because Data FISes may be as long as 8KB, and do not alter internal state of the HBA.

6.1.3 Port Multiplier Errors

When a port multiplier is connected, if a FIS is received from a device, where the PMP field does not match what is expected, the HBA returns R_OK and sets PxIS.IPMS. The HBA shall discard the packet and not update any registers or memory structures based on the FIS contents. In command-based switching, this indicates that the only active PMP port was not properly returned. In FIS based switching, this indicates that a PMP field was returned that is not in the list of active PMPs.

6.1.4 Device Errors

When a FIS arrives which updates the task file, the HBA checks to see if PxTFD.STS.ERR is set. If it is, and PxIE.TFEE is set, the HBA shall generate an interrupt and stop processing any more commands.

6.1.5 Command List Overflow

Command list overflow is defined as software building a command table that has fewer total bytes than the transaction given to the device. On device writes, the HBA will run out of data, and on reads, there will be no room to put the data.

For an overflow on data read, either PIO or DMA, the HBA shall set PxIS.OFS, and if enabled via PxIE.OFE and GHC.IE, generate an interrupt. For an overflow on data writes to DMA, the HBA does not know there is more data until it receives the next DMA Activate. When this occurs, it may optionally set PxIS.OFS and attempt to terminate the transfer. However, this is a fatal condition, and an HBA is allowed to hang on the transfer. For PIO writes, the HBA receives the PIO Setup FIS and therefore knows the length, and therefore may optionally set PxIS.OFS. However, by not satisfying the length, the transfer shall end in an error, and software must recover. Therefore setting PxIS.OFS is optional for both DMA and PIO write conditions.

6.1.6 Command List Underflow

Command list underflow is defined as software building a command table that has more total bytes than the transaction given to the device.

For data writes, both PIO and DMA, the device shall detect an error and end the transfer. These errors are most likely going to be fatal errors that will cause the port to be restarted. For data reads, the HBA shall update its PRD byte count with the total number of bytes received from the last FIS, and may be able to continue normally, but is not required to.

6.1.7 Queued Command Tag Errors

The HBA does not actively check incoming DMA Setup FISes to ensure that the PxSACT register bit for that slot is set.

The reason for this is if the device gives an incorrect tag, it could just as likely be for a tag that is active. In this case, the HBA would see no error, although the data transfer that occurs is incorrect. Therefore, there is little benefit in the HBA checking for inactive tags. Just as in the wrong active tag case, the data transfer that occurs will be incorrect.

Existing error mechanisms, such as host bus failure, or bad protocol, are used to recover from this case.

6.2 Error Recovery

6.2.1 HBA Aborting a Transfer

When the HBA detects an error that it cannot recover from, it may need to end the transfer on the SATA interface.

To do this, the HBA asserts SYNC Escape to stop the bad FIS, and when the device is quiescent, returns to idle. The SATA device should send a D2H Register FIS at this point, with the ERR bit set to indicate an error in the transfer.

When aborting a transfer, the HBA does not wait for the D2H Register FIS before proceeding with error recovery (such as setting interrupt status bits and generating interrupts). This is because a device may be in a hung condition and cannot generate the D2H Register FIS.

6.2.2 Software Error Recovery

When an interrupt is generated due to an error condition, software will attempt to recover. Fatal errors (signified by the setting of PxIS.HBFS, PxIS.HBDS, or PxIS.IFS) will cause the HBA to enter the H:NotRunning state, and clear PxCMD.CR. In this state, the HBA shall not issue any new commands, and, nor acknowledge DMA Setup FISes to process any queued commands. To recover, the HBA must be restarted. For non-fatal errors (signified by the setting of PxIS.INFS or PxIS.OFS) the HBA continues to operate.

If the transfer was aborted (see section 6.2.1), the device is expected to send a D2H Register FIS with PxTFD.STS.ERR set and PxTFD.STS.BSY cleared. Under this scenario, system software knows that the device is in a stable state and transfers may be restarted.

For fatal errors, software must determine which commands were not processed and either re-issue them or notify higher level software that the command failed. The steps involved are listed in the following sections.

6.2.2.1 Non-Queued Error Recovery

The flow for system software to recover from an error when non-queued commands are issued is as follows:

- Reads PxCI to see which commands are still outstanding.
- Reads PxIS.CCS to determine the slot that the HBA was processing when the error occurred.
- Clears PxCMD.ST to reset the PxCI register
- Clears any error bits in PxSERR to enable capturing new errors, and then clearing error bits in PxIS.
- Re-enables PxCMD.ST when the HBA clears PxCMD.CR.
- If PxTFD.STS.BSY was set, before restarting the controller, issue a COMRESET to the device to put it in an idle state.
- Optionally issue a command to recover from the error, for example READ LOG EXT.

Software then either completes commands that did not finish with error to higher level software, or re-issues them to the device.

6.2.2.2 HBA Assisted Queued Error Recovery

The flow for system software to recover from an error when non-queued commands are issued is as follows:

- Reads PxCI to see which commands have not been issued.
- Reads PxSACT to see which commands have been issued but have not yet completed.
- Clears PxCMD.ST to reset the PxCI and PxSACT registers.
- Clears any error bits in PxSERR to enable capturing new errors, and then clearing error bits in PxIS.
- Re-enables the PxCMD.ST bit when the HBA clears PxCMD.CR.
- If PxSACT was non-zero when read, software must perform a READ LOG EXT command to determine the error or perform a COMRESET. If PxTFD.STS.BSY is set, software issues a COMRESET to the device to put it in an idle state.

Software then either completes commands that did not finish with error to higher level software, or re-issues them to the device.

6.2.2.3 Recovery of Spurious COMINIT

If the HBA receives a COMINIT during normal operation, it is considered problematic (i.e. the device could have been changed underneath the command list). When this occurs, the HBA shall halt execution until PxIS.PCS is cleared (this bit is cleared by clearing PxSERR.DIAG.X).

This event may be or may not be an error, and the software recovery listed above may not be necessary. As a general practice, software should ensure PxSERR.DIAG.X is cleared before issuing commands.

7 Hot Plug Operation

7.1 Platforms that Support Cold Presence Detect

For platforms that support cold-presence detect, additional logic is required on the board that implements the SATA ports. How this logic is implemented is beyond the scope of this specification.

7.1.1 Device Hot Unplugged

When a powered-down device is removed, the HBA shall be notified through an external pin for that port going to a logical ‘0’. The HBA set PxIS.CDS to indicate the device changed state. If PxIE.CDE and GHC.IE(z) are set, the HBA shall also generate an interrupt.

7.1.2 Device Hot Plugged

When a device is added, the HBA shall be notified through an external pin for that port going to a logical ‘1’. The HBA shall report that the device status changed by setting PxIS.CDS to indicate the device changed state. If PxIE.CDE and GHC.IE(z) are set, the HBA shall also generate an interrupt.

7.2 Platforms that Support Interlock Switches.

For platforms that support interlock switches, additional logic is required on the board that implements the SATA ports. How this logic is implemented is beyond the scope of this specification. The net result of the logic, though, is that whenever the logic changes state, a high logic level will be sent to the HBA, which shall set PxIS.DIS.

Setting of this bit in and of itself does not imply a hot plug or unplug event, but is a notification to software that the device state may have changed.

7.3 Platforms that Do Not Support Cold Presence Detect nor Interlock Switches

Without the support of cold presence detect or interlock switches, notification of a connection is through COMINIT. When this occurs, the HBA shall generate a COMRESET. The result of the COMRESET sequence shall result in PxSSTS.DET being changed to a non-zero value.

Notification of an unplug event, though, is not supported. AHCI HBAs only set PxSERR.DIAG.X in response to a COMINIT. When a device is removed, commands will time-out. The reason for this is if the interface is in a partial or slumber state when a device is removed, the HBA may not be notified as the interface is in a state that cannot detect the change of presence. As such, the PxSSTS.DET field shall not be updated, and no interrupt shall be generated.

As such, while a disconnection can sometimes be signaled, it is not reliable. To ease hardware implementation, therefore, no additional logic is required to find a disconnection.

7.4 Interaction of the Command List and Port Change Status

Software may have one or several commands in the command list at the time a device is unplugged. A new device may also be inserted before software has had an opportunity to see the change.

It is expected that a hot plug event will mean software will want to stop any outstanding commands and to issue new commands (such as to query a newly attached device)

To accomplish this, an HBA shall stop transferring data, return to an idle condition, and clear PxCMD.CR whenever PxIS.PCS is set. This allows software to see what commands are outstanding by checking PxCI, PxSACT, and PxIS.CCS. Once software has determined which commands need to be re-issued, it shall clear PxCMD.ST and restart the controller by setting PxCMD.ST and taking any necessary actions to enable the SATA device.

8 Power Management Operation

8.1 Introduction

This section covers operations of the HBA and the SATA wire. This specification does not cover any power management that an SATA device may do internally that is transparent to the interface.

8.2 Power State Mappings

The PCI specification defines power management states for devices, which shall be applied to the SATA HBA. They are:

- D0 – working (required)
- D1 – not defined for storage HBAs
- D2 – not defined for storage HBAs.
- D3 – very deep sleep (required). This state is split into two sub-states, $D3_{HOT}$ (can respond to PCI configuration accesses) and $D3_{COLD}$ (cannot respond to PCI configuration accesses). These two sub-states are considered the same, where $D3_{HOT}$ has V_{CC}, but $D3_{COLD}$ does not.

SATA devices may also have multiple power states. Each of these device states are subsets of the HBA's D0 state. They are:

- D0 – Device is working and instantly available.
- D1 – Device enters when it receives a "STANDBY IMMEDIATE" command. Exit latency from this state is in seconds
- D2 – Not currently defined for SATA devices.
- D3 – device enters when it receives a "SLEEP" command. Exit latency from this state is in seconds.

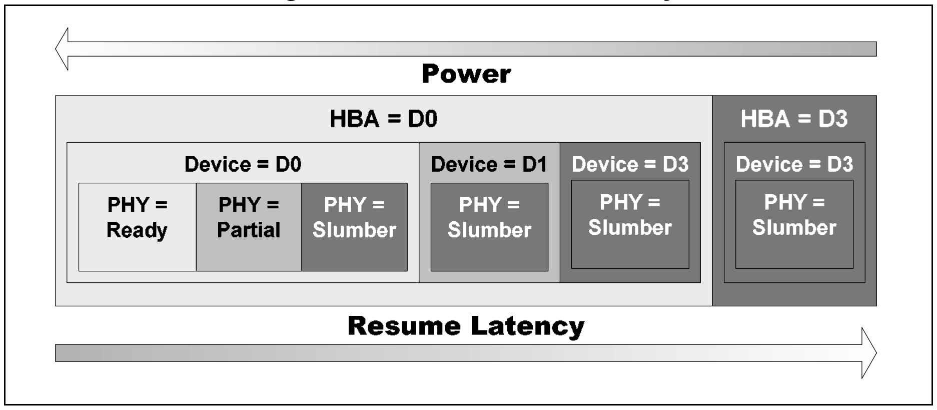
Finally, SATA defines three PHY layer power states. They are:

- PHY Ready – PHY logic and PLL are both on and active
- Partial – PHY logic is powered, but in a reduced state. Exit latency is no longer than 10µs
- Slumber – PHY logic is powered, but in a reduced state. Exit latency can be up to 10ms.

Since these states have much lower exit latency than the D1 and D3 states, AHCI defines these states as sub-states of the device D0 state.

The following picture gives a hierarchical view of power states of SATA.

Figure 17: Power State Hierarchy



8.3 Power State Transitions

8.3.1 Partial and Slumber State Entry/Exit

The partial and slumber states are viewed as a cheap and easy mechanism to save interface power when the interface is idle. It would be most analogous to PCI CLKRUN# (in power savings, not in mechanism).

These states can be entered by software, HBA acceleration, or by the device, via primitives on the SATA interface. Any request can be accepted (PMACK) or rejected (PMNACK) based upon settings in the device and HBA.

8.3.1.1 Device Initiated Entry

The device is enabled to send these primitives through settings in the IDENTIFY DEVICE (ATA) or IDENTIFY PACKET DEVICE (ATAPI) commands, which identifies the capability and the SET FEATURES command, which enables the capability. The capability must be enabled by software. A device may aggressively enter these states while commands are active. An example would be the time from accepting a command FIS until the first data transfer. It could be several milliseconds for the device to seek, so it may request a transition to the partial state while this occurs.

The HBA disposes these requests based upon the PxSCTL.IPM field.

8.3.1.2 System Software Initiated Entry

PxCMD.ICC is used for software initiation of the primitives. Transitions to lower power states on the interface are most likely to be initiated when the device is going to be placed into a D1 or D3 state. Transitions to the active state are most likely used when software knows it has a command to build. It may initiate the transition (by setting PxCMD.ICC to “1h”), and then build the command. If the interface was in the partial state, by the time the PxCI bit is set, the interface should be active. Thus, power savings can be achieved with no additional command latency.

8.3.1.3 HBA Initiated Entry

In some platforms, system software initiated interface power management may not be aggressive enough. It may take a long time for software to recognize an idle condition, so an unacceptable amount of time could pass where no commands are pending but the interface is active. To allow much more aggressive interface power management, AHCI provides the PxCMD.ALPE and PxCMD.ASP fields.

When PxCMD.ALPE is set, whenever the HBA recognizes that there are no commands to process, it shall initiate a transition to partial or slumber based upon the setting of PxCMD.ASP. The HBA recognizes no commands to transmit as either:

- The PxSACT register is zero, and the HBA updates the PxCI register from non-zero to zero. This would occur when the HBA is operating with non-queued commands, and the last command active just cleared its BSY bit.
- The PxCI register is zero, and a Set Device Bits FIS arrives which updates the PxSACT field from non-zero to zero. This would occur when the HBA is operating on queued commands, and the last active command to the device is marked as finished with the Set Device Bits FIS.

If the PxSACT and PxCI registers are both cleared, and the interface is in an active state, the HBA takes no action to aggressively put the interface into a lower power state. This is recognized as a condition where software may be bringing the interface to an active state to start placing new commands in the command list. If the HBA were to take action to put the interface into partial or slumber, unacceptable latency may occur for the command.

8.3.1.4 HBA Initiated Exit

Before performing any data transfer on the link, the HBA must first ensure the link is in an active state. If the link is in an inactive state, a COMWAKE must be issued, and the HBA must wait until the link is in an active state before proceeding.

8.3.2 Device D1, D3 States

These states are entered when system software has determined that no commands will be sent to this device for some time. To enter these states, software must perform two actions. The first is to issue the command to the device (STANDY IMMEDIATE for D1, SLEEP for D3), and the second step is to put the interface into a slumber state (setting PxCMD.ICC to “6h”). Note that the interface should already be in a slumber state, initiated by the device, as a result of the device being placed into D1 or D3. However, it is still good practice for software to request the HBA to enter this state if the device does not.

8.3.3 HBA D3 state

After the interface and device have been put into a low power state, the HBA may be put into a low power state. This is performed via the PCI power management registers in configuration space. AHCI only supports the D3 state.

There are two very important aspects to note when using PCI power management.

When the power state is D3, only accesses to configuration space are allowed. Any attempt to access the register memory space must result in master abort.

When the power state is D3, no interrupts may be generated, even if they are enabled. If an interrupt status bit is pending when the controller transitions to D0, an interrupt may be generated.

AHCI does not specify how power is saved in the HBA in these low power states. Actions such as clock stopping or power plane control are left to each implementation.

8.4 PME

When the HBA is in the D3 state, it may optionally wake based on a change in the device state. Since the device is in a sleep state and the interface is in slumber when the D3 state is entered, the only change in device state that is recognized is a hot plug event.

PME must be generated when in D3 under the following conditions:

- PxIS.PCS set due to a device connection
- PxIS.DIS set, indicating an interlock switch has been opened or closed (for platforms that support interlock switches)
- PxIS.CPDS set, indicating cold presence state change of a device (for platforms that support cold presence detect.)

If any of these bits are set, regardless of the setting of the enables in PxIE and GHC.IE, the HBA shall generate PME#.

8.5 Capability / Control Interaction for Link Partial / Slumber State

There are several bits that software must use to enable transitions to the partial and slumber states. This section describes the precedence and interactions among those bits.

The first bits software must use are CAP.SSC and CAP.PSC. These bits indicate whether the HBA is capable of handling partial or slumber link transitions, either as an initiator or a target. If either or both of these bits are cleared, then software must not allow the HBA to initiate requests to this state through the PxCMD.ICC field, nor by enabling aggressive link power management via PxCMD.ALPE / PxCMD.ASP. If an HBA is enabled to initiate these transitions, but is not capable, it is allowed to go into an unrecoverable state.

Additionally, if the HBA is not capable of one or both of these transitions, requests from the device must be disallowed. This can be accomplished by preventing the device from asking for these transitions, or by setting the PxSCTL.IPM field to NAK transitions initiated by the device. If the HBA receives a request for these states and is not capable, it must NAK the request.

Transitions to low power link state may occur concurrently. For example, the device may be programmed to request a transition to partial or slumber when it is busy and cannot perform a data transfer, and the HBA may be programmed to automatically enter partial/slumber (through PxCMD.ALPE) when no commands are active. Even with these two activities in place, software may still find it advantageous to force the link to partial/slumber through PxCMD.ICC.

The Serial ATA 1.0a specification describes the interaction of the link layer when a host initiated and device initiated power state transition occur concurrently, and therefore this is not discussed here. Additionally, there is no priority between PxCMD.ICC and PxCMD.ALPE.

9 Port Multiplier Support

Port Multiplier support for HBAs is optional, and its support is indicated to system software via CAP.PMS. If Port Multipliers are supported in the HBA, it must support command-based switching. A higher level of performance may be supported via FIS-based switching, which is indicated by CAP.PMFS also being set. Table 1 indicates to system software the level of Port Multiplier support:

Table 1: HBA Port Multiplier Support Level

CAP.PMS	CAP.PMFS	Port Multiplier Support Level
'0'	N/A	No Port Multiplier support
'1'	'0'	Command-based switching Port Multiplier support
'1'	'1'	Command-based and FIS-based switching Port Multiplier support.

This section describes the hardware and software differences necessary to make a Port Multiplier work.

9.1 Command Based Switching

In this mode of operation, a communication path is opened between the HBA and a device through the Port Multiplier. Since Port Multipliers are meant to be simple, the burden of making a connection is on AHCI software, to ensure that multiple commands are not outstanding to different devices behind the Port Multiplier.

9.1.1 Non-Queued Operation

When running non-queued commands, the command list may be filled with any combination of ports, and each command list entry can target a different port. There is no fixed relationship between number of commands allocated to the device and the number of devices behind a Port Multiplier. For example, 29 commands could be allocated to the device behind PM port #0,and 3 commands for the remaining devices, if that is desired by system software.

Since the commands are non-queued, the HBA shall execute each command entry in its entirety before moving to the next entry in the command list, which may include a command to a different port.

This places special burden on system software for building commands. Since the HBA operates in order in its command list, system software must not fill the list in sequential order with commands to a single device; otherwise another device may get starved. Take the following example:

- 4 devices attached to a PM behind a port
- The operating system presents several commands
  - 4 to the device behind PM port #0
  - 2 to the device behind PM port #1
  - 1 to the device behind PM port #2

If the AHCI software placed these commands in sequential order, starting at command slot 0, the list would look as follows:

- Slot #0: PM Port #0, Command 1
- Slot #1: PM Port #0, Command 2
- Slot #2: PM Port #0, Command 3
- Slot #3: PM Port #0, Command 4
- Slot #4: PM Port #1, Command 1
- Slot #5: PM Port #1, Command 2
- Slot #6: PM Port #2, Command 1

This would cause the device behind port #2 to not execute its command for a very long time. In the worst case scenario, 31 commands to other devices behind different PM ports may execute before this command is allowed to execute.

A better placement of commands to result in fairer execution would be as follows:

- Slot #0: PM Port #0, Command 1
- Slot #1: PM Port #1, Command 1
- Slot #2: PM Port #2, Command 1

- Slot #8: PM Port #0, Command 2
- Slot #9: PM Port #1, Command 2
- Slot #16: PM Port #0, Command 3
- Slot #24: PM Port #0, Command 4

Mapping commands in this fashion helps ensure that the device behind PM port #2 has a fairer chance of executing. The above algorithm results in round-robin execution of commands. Many algorithms for fairness exist, and their implementations are beyond the scope of this specification.

In order to find the best slot for placing the next command, software should use PxCMD.CCS to find the current slot the HBA is executing, and place the command in an appropriate slot for the fairness algorithm that is desired.

9.1.2 Queued Operation

When running queued commands, system software must take extra care. Since queued commands result in two different operations (command issue, clear of BSY, then data transfer), if commands were sent to different ports, the Port Multiplier may issue FISes back to the HBA in an interleaved manner. This will break an HBA that only supports command-based switching.

Therefore, when executing a queue, system software must only add commands to the command list that target a single port behind the multiplier, wait for the commands to finish (PxSACT bits all cleared), then add commands for a different port. Additionally, the tags used must match the command slot entries.

9.2 FIS Based Switching

This is reserved for a future version of the AHCI specification.

10 Port Selector Support

An HBA may optionally support the COMRESET sequence described in the Port Selector specification for activating a port. Support for this feature is indicated via CAP.PSSA being set to ‘1’.

For HBAs that support this feature, the following bits shall exist for each implemented port:

- PxCMD.PSA – Starts/stops the COMRESET sequence.
- PxIS.PCS – Interrupt status bit that is set upon the completion of the sequence.
- PxIE.PCE – Enables interrupt generation when the sequence is complete.

To start the sequence, software sets PxCMD.PSA. The HBA starts the COMRESET sequence. Upon receiving COMINIT, the HBA sets PxIS.PCS (a reflection of PxSERR.DIAG.X), and if enabled, an interrupt shall be generated. Additionally, PxCMD.PSA shall be cleared by the HBA.

If the sequence does not complete successfully, the HBA attempts the sequence again. This continues until either the sequence completes successfully, or software clears PxCMD.PSA.

11 Enclosure Management Services

An HBA supports enclosure management as specified in the "Serial ATA II: Extensions to Serial ATA 1.0" specification. An HBA that integrates enclosure management logic shall allocate one of its 32 ports for use as the enclosure management port. It reports a signature of enclosure management as per that specification, and software shall use the command list for this port to execute enclosure management commands per that specification. The enclosure management logic can either be internal or external to the HBA.

While allocating a port for enclosure management reduces the available ports on an HBA to 31 from a maximum of 32, constructing enclosure management services in this way reduces software complexity. Software may talk to an enclosure management processor in the same fashion, regardless of whether that processor is integrated in the HBA, or externally, such as through a Port Multiplier.

12 Platform Communication

12.1 Software Initialization of HBA

Before any useful work can be performed, the HBA must be initialized. Initialization consists of two independent phases: a firmware phase (platform BIOS or on-board ROM) and a system software phase. Initialization of the HBA's PCI configuration space is beyond the scope of this specification.

12.1.1 Firmware Specific Initialization

To aid system software during runtime, the BIOS shall ensure that the following registers are initialized to values that are reflective of the capabilities supported by the platform. Firmware shall always initialize these values:

- CAP.SCD
- CAP.SIS
- CAP.SSS

Additionally, firmware shall also insure that the following registers are initialized:

- PI
- PxCMD.HPCP. Firmware shall initialize the HPCP register for each port implemented on the platform (as defined by the PI register).

After firmware has initialized the aforementioned registers, it shall then perform the following steps to complete the staggered spin-up process (if applicable to the platform) on each port implemented by the AHCI HBA (as indicated by the PI register):

1. Ensure that PxCMD.ST and PxCMD.CR are '0', and PxSCTL.DET = '0h'.
2. Initiate a spin up of the SATA drive attached to the port; i.e. set PxCMD.SUD to '1'.
3. Wait for a positive indication that a device is attached to the port (the maximum amount of time to wait for presence indication is specified in the *Serial ATA: High Speed Serialized AT Attachment v1.0a* specification). This is done by an polling PxSSTS.DET. If these bits return a value of '3' when read, then system software shall continue to step 4, otherwise it moves to the next port and returns to step 1.
4. Wait for indication that SATA drive is ready. This is determined via an examination of PxTFD.STS. If PxTFD.STS.BSY, PxTFD.STS.DRQ, and PxTFD.STS.ERR are all '0', prior to the maximum allowed time as specified in the ATA/ATAPI-6 specification,, the device is ready.

12.1.2 System Software Specific Initialization

This section describes how system software places the AHCI HBA into a *minimally* initialized state. Because the term *system software* is used to collectively reference both the platform BIOS (including option ROM) and operating system, this section applies to any software that is AHCI aware.

To place the AHCI HBA into a minimally initialized state, system software shall:

1. Indicate that system software is AHCI aware by setting GHC.AE to '1'.
2. Determine which ports are implemented by the HBA, by reading the PI register. This bit map value will aid software in determining how many ports are available and which port registers need to be initialized.
3. Ensure that the controller is not in the running state by reading and examining each implemented port's PxCMD register. If PxCMD.ST and PxCMD.CR are both cleared, the port is in an idle state. Otherwise, the port is not idle and should be placed in the idle state prior to manipulating any HBA and port specific registers. System software places a port into the idle state by clearing PxCMD.ST and waiting for PxCMD.CR to return '0' when read. Software shall wait a maximum of 500ms for this to occur.
4. Determine how many command slots the HBA supports, by reading CAP.NCS.
5. For each implemented port, system software shall allocate memory for and program:
   - PxCLB and PxCLBU (if CAP.S64A is set to '1')
   - PxFB and PxFBU (if CAP.S64A is set to '1')

It is good practice for system software to 'zero-out' the memory allocated and referenced by PxCLB and PxFB.

6. For each implemented port, clear the PxSERR register, by writing '1s' to each implemented bit location.
7. Determine which events should cause an interrupt, and set each implemented port's PxIE register with the appropriate enables. To enable the HBA to generate interrupts, system software must also set GHC.IE to a '1'.

   Note: Due to the multi-tiered nature of the AHCI HBA's interrupt architecture, system software must always ensure that the PxIS (clear this first) and IS.IPS (clear this second) registers are cleared to '0' before programming the PxIE and GHC.IE registers. This will prevent any residual bits set in these registers from causing an interrupt to be asserted.

At this point the HBA is in a minimally initialized state. System software may provide additional programming of the GHC register and port PxCMD and PxSCTL registers based on the policies of the operating system and platform – these details are beyond the scope of this document.

12.2 Software Manipulation of Port DMA Engines

Each port contains two major DMA engines. One DMA engine walks through the command list, and is controlled by PxCMD.ST. The second DMA engine copies received FISes into system memory, and is controlled by PxCMD.FRE.

12.2.1 Start (PxCMD.ST)

When PxCMD.ST is set, software is limited in what actions it is allowed to perform on the port.

- It shall not manipulate PxCMD.POD to power on or off a device through cold presence detect logic (if supported by the HBA).
- It shall not manipulate PxSCTL.DET to change the PHY state
- It shall not manipulate PxCMD.SUD to spin-up the device (if supported by the HBA)
- It shall not perform Port Selector activation through PxCMD.PSA (if supported by the HBA)

The above actions are only allowed while the HBA is idle, indicated by both PxCMD.ST and PxCMD.CR being equal to '0'. This is noted by the HBA state machine H:NotRunning state. If software performs any of the above actions while the port is not idle (PxCMD.ST or PxCMD.CR set), indeterminate results may occur.

Software shall not write PxCMD.ST to '1' from a '0' until it sees that PxCMD.CR is '0'.

12.2.2 FIS Receive Enable (PxCMD.FRE)

When PxCMD.FRE is set, the HBA receives FISes from the device and copies them into system memory. When PxCMD.FRE is cleared, received FISes are held internally, and if the HBAs internal FIFO is full, further FIS reception is blocked.

Software is allowed to manipulate this bit so that it may move the FIS receive are to a new location. When this bit is cleared, software must first wait for PxCMD.FR to clear, indicating that the DMA engine for FIS reception is in an idle condition.

Software shall not clear this bit while PxCMD.ST remains set.

Upon HBA reset, this bit is cleared. The D2H register FIS containing the device signature shall be accepted by the HBA, and the signature field updated.

12.3 Reset

AHCI supports three levels of reset:

- Device – a single device on one of the ports is reset, but the HBA and physical communication remain intact. This is the least intrusive.
- Port – the physical communication between the HBA and device on a port are disabled. This is more intrusive
- HBA – the entire HBA is reset, and all ports are disabled. This is the most intrusive.

When a reset occurs, communication shall be re-established to the device through a reset. At the end of this process, the device, if working properly, shall send a D2H Register FIS, which contains the device signature. When the HBA receives this FIS, it updates PxTFD.STS and PxTFD.ERR register fields, and updates the PxSIG register with the signature.

12.3.1 Device Reset

Legacy software contained two mechanisms for generating a reset to an SATA device – by setting the SRST bit in the device control register, and by sending a device reset command (command code 08h).

AHCI supports this mechanism via software building a reset command and setting the CHz[R] bit. When this bit is set, the HBA shall perform a SYNC escape if necessary to put the device into an idle condition before sending the command, and it shall set PxTFD.STS.BSY

When using this command, software must ensure that it is the only command in the list. It must clear PxCMD.ST, wait for the port to be idle (PxCMD.CR = ‘0’), and then re-set PxCMD.ST.

12.3.2 Port Reset

If a port is not functioning properly after a device reset, software may attempt to re-initialize communication with the port via a hard reset. It must first clear PxCMD.ST, and wait for PxCMD.CR to clear before re-initializing communication. However, if PxCMD.CR does not clear within a reasonable time (500 ms), it may assume the interface is in a hung condition and may continue below.

Software resets the port by writing a “1h” to the PxSCTL.DET field to put the interface into a reset state, then after a reasonable time (10 ms), clear PxSCTL.DET to “0h”..

When PxSCTL.DET is set to 1h, PxTFD shall be reset to 7Fh by the HBA. When PxSCTL.DET is set to 0h, upon receiving a COMINIT from the attached device, PxTFD.STS.BSY shall be set to ’1’ by the HBA.

12.3.3 HBA Reset

If the HBA becomes unusable for multiple ports, and a device reset or port reset does not correct the problem, software may reset the entire HBA via GHC.HR. Software sets the bit, and the HBA performs an internal reset action. The bit is cleared to ‘0’ by the HBA when the reset is complete. A software write of ‘0’ has no effect. Software sets GHC.HR and polls until the bit is a ‘0’, at which point it knows that the reset has completed.

If the HBA has not reset itself and cleared GHC.HR within 1s, the controller is considered in a hung or locked state, and higher level software must take action.

When GHC.HR is set, GHC.IE, the IS register, and all port registers in the HBA’s register memory space are reset. The HBA’s configuration space is not affected by the setting of GHC.HR. If the HBA supports staggered spin-up, no further action is taken on the port – software must spin-up the device. If the HBA does not support staggered spin-up, a COMRESET is sent on the port.

12.4 Interface Speed Support

The HBA indicates the maximum speed it can support via the CAP.ISS register. Software can further limit the speed of a port by manipulating each port's PxSCTL.SPD field to a lower value. If software writes a value that is greater than the value in CAP.ISS, the actual speed negotiated will be limited by CAP.ISS, as shown in the following table:

CAP.ISS	PxSCTL.SPD	Maximum Speed Negotiated
1h	0h	1.5 Gbps
1h	1h	1.5 Gbps
1h	2h	1.5 Gbps
2h	0h	3 Gbps
2h	1h	1.5 Gbps
2h	2h	3 Gbps

12.5 Interaction of PxSCTL.DET and PxCMD.SUD

The PxSCTL.DET value manipulates the behavior of the PHY. For platforms that support staggered spin-up, PxCMD.SUD also manipulates the behavior of the PHY. The following table describes the interaction.

PxSCTL.DET	PxCMD.SUD	Mode	Behavior
0h	0	Listen	Interface in a reduced power state. • COMINIT received, PxSERR.DIAG.X set, no response sent • COMWAKE ignored. Software shall only place the port into this state when it believes that no device is connected on the port. In this mode, the HBA forces the link into a low power state without requesting a partial/slumber transition.
0h	0 → 1	Spin-Up	HBA shall send COMRESET, begin initialization sequence
0h	1	Normal	**Normal Mode:** This is the state where the ICH6 is performing data transfers. • COMINIT received, PxSERR.DIAG.X set, send COMRESET, begin initialization sequence • COMWAKE received, wake the link
1h	0	Illegal	If software programs this condition, indeterminate results may occur.
1h	1	Reset	HBA ontinuously transmits COMRESET. Do not listen for COMINIT.
1h → 0h	1	Initialize	Stop sending COMRESET, begin initialization sequence.
4h	N/A	Off	Off

Software must only clear PxCMD.SUD if it believes that no device is attached. In Listen Mode (PxSCTL.DET = 0h and PxCMD.SUD = 0), the HBA is allowed to place the link into a reduced power state, equivalent to the power consumed while in the slumber state.

It does this without first negotiating for the slumber state, as asking for a transition to slumber when no device is attached will fail, and therefore the link would stay in a high power state.

If, however, software forces Listen Mode without first checking for device presence, an attached device would see the link entering a low power state without a negotiation, and its response will most likely be to send a COMINIT. As a COMINIT shall set PxSERR.DIAG.X and generate an interrupt, software shall have no choice but to spin up the device, thinking a new device was inserted.

Therefore, software shall perform the following sequence prior to clearing PxCMD.SUD.

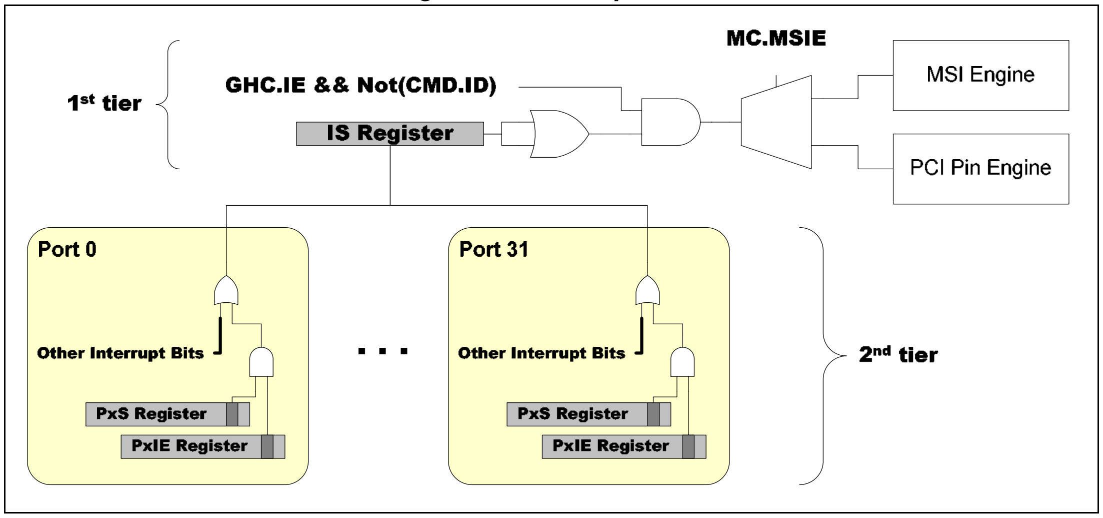
1. Write PxCMD.ICC to "01h" to move the port to an active state.
   Check PxSSTS.DET. If communication has been established (DET = 3h) a device is attached,

12.6 Interrupts

12.6.1 Tiered Operation

AHCI defines a two-tiered interrupt architecture, which allows for reporting of interrupts to software such that software can handle interrupts through the least amount of programming and status checking. A diagram of the interrupt tiers is shown below:

Figure 18: Interrupt Tiers



12.6.1.1 First Tier (IS Register)

The first tier is identified by the GHC and IS registers.

GHC.IE register enables interrupts for the entire HBA. This is the ‘master enable. Until this bit is set, the HBA shall not generate any interrupts. When it is cleared, the HBA may generate interrupts. This bit is only a mask, and does not affect the setting of any of the interrupt status bits in any of the ports. These bits are set regardless of whether or not interrupts are enabled.

The 32-bit IS register reports whether a port has an interrupt pending. This is a bit-mapped register indicating a bit for each of the 32 ports allowed in AHCI. Each bit location can be thought of as reporting a ‘1’ if the virtual “interrupt line” for that port is indicating it wishes to generate an interrupt. That is, if a port has one or more interrupt status bit set, and the enables for those status bits are set, then this bit shall as set. The bits in this register are read/write clear. It is set by the level of the virtual interrupt line being a set, and cleared by a write of ‘1’ from the software.

This register allows software to perform a quick glance of the HBA to see which ports are reporting interrupts. By being read/write clear, it also allows for the implementation of message signaled interrupts.

12.6.1.2 Second Tier (PxIS Registers)

The second tier is identified in each port, through the PxIS (status) and PxIE (interrupt enable) registers. The PxIS register for each port has various interrupt bits

Each one of these interrupts can be individually enabled or disabled for a port, by setting the corresponding bit in PxIE. The status bit in PxIS is always set regardless of the setting of the corresponding PxIE bit.

12.6.2 HBA/SW Interaction

12.6.2.1 Pin Based and Single MSI Message Based Behavior

This is the mode of interrupt operation if any of the following conditions are met:

- MSI is disabled via MSICAP.MSIE
- HBA only supports a single MSI interrupt via the configuration register MSICAP.MCC
- HBA only enabled for a single interrupt via MSICAP.MEE
- MSICAP.MEE is programmed to a larger value than MSICAP.MCC

In this mode, the IS register determines whether the PCI interrupt line shall be driven active or MSI message shall be sent. The PxIS register contains status information to generate the interrupt. The resulting logical "AND" of the PxIS bit and corresponding PxIE bit results in a "virtual wire" that sets a sticky bit in the IS register. When a level of a virtual wire for a port is '1', the IS register bit for that port shall be set.

The resulting output of the IS register is sent to one of two places. If MSIs are not enabled, the resulting IS bits are logical "ORed" together – any bit being a one causes the PCI interrupt line to be active (electrical '0'). If MSIs are enabled, any change to the IS register that doesn't result in the register being all cleared shall cause an MSI to be sent. Therefore, that in wire mode, a single wire remains active, while in MSI mode, several messages may be sent, as each edge triggered event on a port shall cause a new message, as shown in Table 2.

In order to clear an interrupt, software must first clear the event from the PxIS register, then clear the interrupt event from the IS register. If software clears IS register only, leaving the level of the virtual wire from the PxIS register set, the resulting level of '1' shall cause the IS register bit to be re-set.

Table 2: MSI vs. PCI IRQ Actions

Status of Tier 1 Register	Wire-Mode Action	MSI Action
All bits '0'	Wire inactive	No action
One or more bits set to '1'	Wire active	New message sent
One or more bits set to '1', new bit gets set to '1'	Wire active	New message sent
One or more bits set to '1', software clears some (but not all) bits	Wire active	New message sent
One or more bits set to '1', software clears all bits	Wire inactive	No action

12.6.2.2 Multiple MSI Based Messages

An HBA may optionally support multiple MSI messages for better performance. In this mode, each port has its own interrupt message. To support this mode, the MSICAP.MCC field represents a power-of-2 wrapper on the number of implemented ports in the global memory space PI register. For example, if 3 ports are implemented, then the MSICAP.MCC field must be '010' (4 interrupts).

Multiple-message MSI enables each port to send its own interrupt message, as opposed to a single message for all ports. When enabled for multiple message generation, generation of interrupts is no longer controlled through the IS register nor the GHC.IE interrupt enable. Interrupts are instead controlled within each port.

Each port receives its own interrupt message, up to a maximum of 16 interrupts (16 ports). 16 interrupts is the practical limit of MSI messages in an x86/Windows environment. The mapping of ports to interrupts is done in a 1-1 relationship, up to the maximum number of assigned interrupts. Any ports implemented beyond that point use interrupt message 15.

Take the following example in Table 3, where 8 ports are implemented (ports 0 – 7), and 4 messages are allocated (messages 0 – 3). The mapping of ports to interrupts is as follows:

Table 3: Port/MSI Message Mapping, Example 1

Port	Interrupt Message
0	0
1	1
2	2
3	3
4	
5	
6	
7	

Each implemented port maps to an interrupt message of the same value. For example, port 2 maps to message 2, even if port 1 is not implemented. Below is an example, showing 6 ports, where only ports 0, 3, and 5 are implemented:

Table 4: Port/MSI Message Mapping, Example 2

Port	Interrupt Message
0	0
1	1 (unused)
2	2 (unused)
3	3
4	4 (unused)
5	5
-	6 (unused)
-	7 (unused)

In this example, since the HBA had 6 ports, it is required to ask for 8 interrupt messages. In this example, therefore, messages 6 and 7 are also not used.

When generating an MSI message, a port looks at its PxIS register, and uses the following rules to generate a message:

- If a new bit is set in PxIS, and the corresponding bit in PxIE is set, send a message
- If bits are cleared in PxIS, and other bits remain set, if their corresponding bits in PxIE are set, send a message.

The HBA is allowed to set bits in IS if it wishes. However, it shall not generate any interrupt messages based upon bits in IS being set, and it shall not use GHC.IE being ‘0’ as a gate to generating messages for the port.

12.6.2.3 MSI-X Based Messages

Reserved for a future revision of AHCI. AHCI 1.0 does not support MSI-X generation.

12.6.3 Disabling Device Interrupts (NIEN Bit in Device Control Register)

This bit was part of the device control register (legacy I/O addresses 3f6h for primary, 376h for secondary). A legacy HBA must snoop writes to this bit to know whether to mask device interrupts. An AHCI HBA does not snoop this bit. All masking is controlled via the PxIE register.

If software sets the NIEN bit, the ‘I’ bit on received FISes will not be set, and software loses flexibility on when interrupts are generated by the HBA. It is strongly suggested that AHCI system software not set this device bit and use the PxIE register for interrupt masking.

12.7 Interlock Switch Operation

In systems that support an interlock switch, the platform contains a mechanical mechanism that acts as an attention indicator or locking mechanism. When this mechanism changes state (such as a button pressed or switch opened/closed), a line changes state.

HBAs that support interlock switches have the following additional features:

- CAP.SIS shall be set.
- An additional input pin per port on the HBA.

12.8 Cold Presence Detect Operation

In systems that support cold presence detect, the platform board contains voltage comparator logic, as described in the Serial ATA II specification, for recognizing attachments, and FET control to supply power to the device.

HBAs that support cold presence-detect have the following additional features:

- CAP.SCD shall be set.
- An additional input pin per port on the HBA.
- An additional output pin per port on the HBA to be used as FET control for the power rails to the device.
- For each port, PxCMD.POD shall be read/write.

When an HBA that supports cold presence-detect is first powered-up, no power is supplied to the devices. Initialization software or ROM checks the status of the ports, and if a device is connected, sets PxCMD.POD to a ‘1’ to supply power to the port.

12.9 Staggered Spin-up Operation

In systems that support staggered spin-up, an HBA can individually spin-up implemented SATA ports. In systems that do not support staggered spin-up, the HBA spins-up all SATA port upon receiving power to the HBA.

HBAs that support staggered spin-up have the following additional features:

- CAP.SSS shall be set.
- For each port, PxCMD.SUD will be read/write.

12.10 Activity LED

An HBA optionally drives an output pin that can be connected to an external LED based upon activity of the various ports in the HBA. The intended use of this LED is for desktop and mobile systems that contain only a single LED to indicate device activity. An HBA indicates support for this pin via CAP.SAL.

The intent of this LED output is to replace the DASP functionality that is provided in parallel ATA systems. The DASP logic in the device would light the LED under the following scenarios:

- During boot, soft reset, or device reset: LED would be driven by the slave device on a cable (if present) to indicate slave presence. The LED would remain on until a command was written to the slave device, or 30 seconds, whichever comes first.
- During normal operation, while the BSY bit was set in the task file for ATA commands. It may optionally be on for ATAPI commands (A0h and A1h).

In AHCI, slave devices are not supported – every device is a master. Therefore, the first LED mechanism is not supported by an AHCI HBA. An HBA that supports legacy operation will have to consider this support if it supports master/slave emulation.

To support the second mechanism of LED operation, the HBA shall drive the LED pin active if CAP.SAL is set, and:

- If (PxCI != 0h or PxSACT != 0h) and PxCMD.ATAPI = ‘0’.
- If (PxCI != 0h or PxSACT != 0h) and PxCMD.ATAPI = ‘1’ and PxCMD.DLAE = ‘1’,

When PxCI and PxSACT are both cleared to 0h the LED shall be driven off.

12.11 BIST

BIST is entered when software builds a BIST FIS in the command list and sets CFISz[B]. Once a BIST command is placed into the list, SW is not allowed to build any more commands until it clears PxCMD.ST.

Details of how an HBA operates in the test mode are outside the scope of this specification. A series of vendor specific tests may be performed using vendor specific registers (such as those in the HBA's PCI configuration space). The details of BIST are not defined in AHCI to allow vendors the freedom of constructing either very elaborate or very light testing schemes.

The test mode is exited when system software clears PxCMD.ST and writes a value of 1h into PxCTL.DET. Clearing PxCMD.ST shall put the DMA engines into an idle condition, and writing to PxCTL.DET shall reset the interface.